PC
.7
e Mikrocomputer-Zeitschrift
7 DM · 60 öS · 7 sfr. · Juli 1987

PC-Technik:
Neuer IBM-Systembus
im Detail

Atari-ST:
3D-Darstellung mit GFA-Basic

Software:
Co-Routinen in C
und Turbo-Pascal

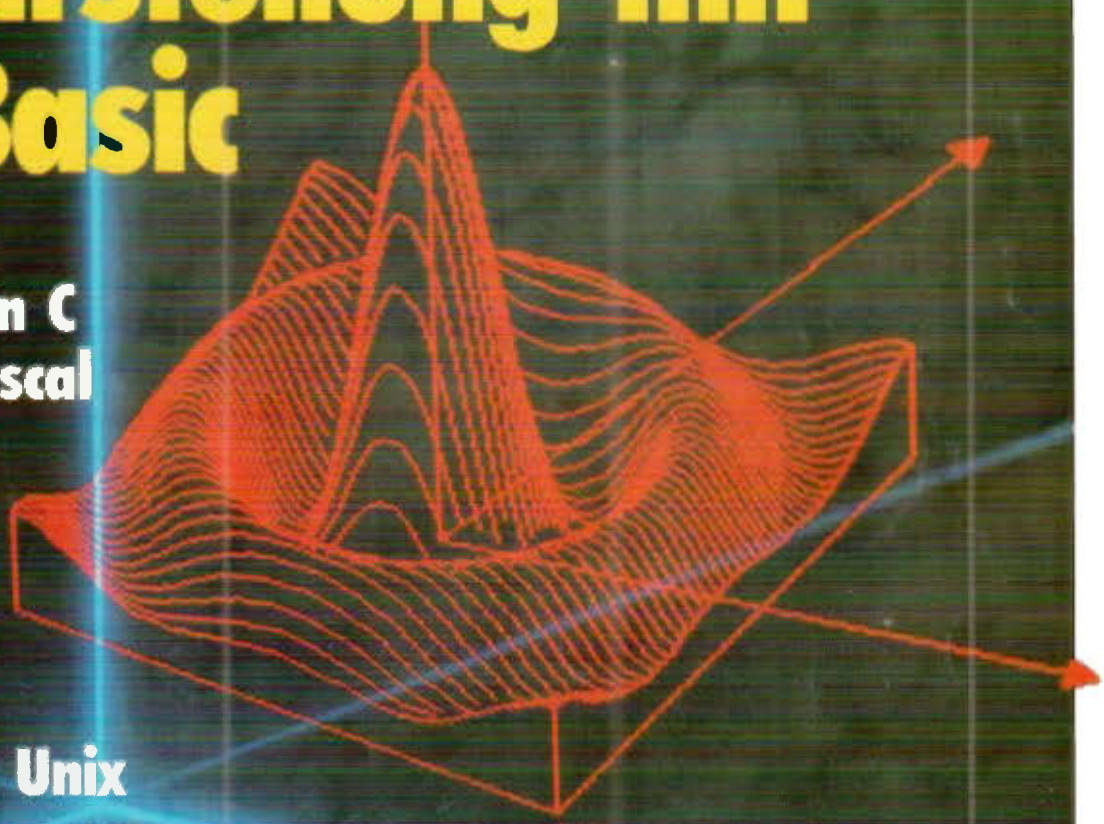
Do what
I mean
mit MuLisp

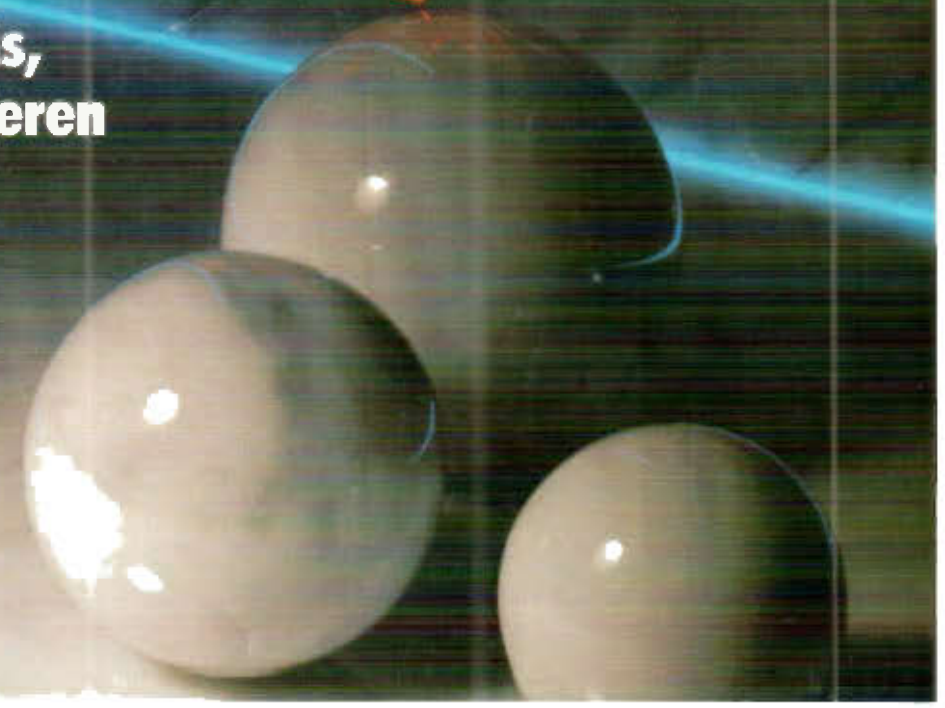
Serien:
Einführung in Unix

PALs, PLAs, EPROMs,
PROMs programmieren

Test:
Preiswertes
CAD-Programm

Im Vergleich:
Tulip-AT, MCI-AT,
Osborne-AT





Liebe Leser!

In diesem Heft gibt es den zweiten Teil zu unserem Allprog-Gerät. Es scheint, als hätten wir international Staub aufgewirbelt, denn es kamen selbst Anfragen aus den USA. Die Zahl der Anrufer aus der Bundesrepublik kann man gar nicht zählen. So ist das eben mit den Bauanleitungen in mc: Wir bemühen uns immer das Beste, Interessanteste und Nützlichste für unsere Leser herauszufinden und zu veröffentlichen. Dabei kommt es uns vor allem auf die gute Zusammenarbeit aller an einem solchen Projekt Beteiligten an. Die Autoren sollen bei uns gut aufgehoben sein. Die Hardwareanbieter sollen mit den Bausätzen durchaus gute Geschäfte machen. Wir aber wollen, daß unsere Bauvorschläge für Sie, für unsere Leser, durch und durch transparent sind. Deshalb werden in diesem Heft die Einzelheiten geschildert, die zur technischen Programmierung der Bausteine benötigt werden. Lesen Sie also zu mc-Allprog den zweiten Teil.

Als erste in Europa schildern wir die technischen Einzelheiten des neuen IBM-Busses. Wir hoffen damit, allen Hardware-Anbietern eine Ahnung darüber geben zu können, ob sich vielleicht die Entwicklung neuer Ergänzungskarten technisch einfach machen läßt. Damit alle, die es interessiert, genügend vorinformiert sind, falls der neue Boom losbrechen sollte. Der Artikel wird zweiteilig sein. In diesem Heft zunächst einmal das Mikro-Kanal-Konzept anhand der Signal-Namen und deren inhaltlicher Bedeutung. Natürlich auch für die geschildert, die allein aus Neugier auf diese Technik gespannt sind. Mit mc sind Sie immer gut informiert.

Daß wir nicht nur IBM-Fans sind, das sollen Ihnen Artikel über den Macintosh beweisen. Der große Macintosh von Apple benutzt einen Bus, der ebenso interessant ist wie der der neuen IBM-Systeme. Aber dazu werden Sie erst im nächsten Heft die technische Einzelheiten erfahren. Hervorzuheben ist, daß Apple uns diese Einzelheiten bekannt gemacht hat und wir diese Kenntnisse an Sie weitergeben wollen, damit Sie Ihren Macintosh besser nutzen können.

Der Biber-Wettbewerb neigt sich für Sie zum Ende. Am 30. 6. 1987 müssen die letzten Biber abgeschickt sein, also genau einen Tag nach dem offiziellen Erstverkaufstag dieser Ausgabe von mc. Für uns beginnt dann die Auswertungsarbeit. Es wird eine harte Arbeit, denn Sie haben viele Biber eingesandt. Mehr darüber bald in mc,

Ihr

Ulrich Rohde

mc-inhalt

mc-editorial

Liebe Leser 3

mc-briefe 6

mc-info 8

Spruch des Monats 58

HP-Expo '87 75

Mit DES Daten schützen 75

Taiwan schlägt zu 101

Altos mit 80386 101

Zitat des Monats 101

mc-bücher 30

mc-soft

3D-Darstellung von Funktionen 40
Der Atari ST prüft unter GFA-Basic jeden Punkt der Darstellung auf Sichtbarkeit, bevor er ihn zeichnet – ein sehr rechenintensives Programm

Programmierumgebung Dwim und Korr 44
Do what I mean (Dwim) kennzeichnet bei Computern eine Benutzeroberfläche, die kleine Eingabe-Fehler toleriert. Der Beitrag zeigt die Realisierung in muLisp

Arithmetische Ausdrücke analysieren 48
Das Turbo-Prolog-Programm, das die üblichen mathematische Schreibweisen „versteht", gibt ein gutes Beispiel für die Arbeit mit einer Grammatik in dieser verbreiteten Programmiersprache

Coroutinen in C und Pascal 51
Modula-2-Programmierern sind sie unentbehrlich: Coroutinen, also aufrufbare Prozeduren, die ihre lokalen Daten nicht vergessen. Ein entsprechendes Programm in C (für ein 68000-System unter Unix) und Turbo-Pascal (für MS-DOS und CP/M) wird hier vorgestellt

Hardcopy von CGA-Farbgrafiken 54
Bei allem, was mc über EGA- und Hercules-Grafik gebracht hat, ist die CGA-Karte nicht in Vergessenheit geraten: Das Programm sorgt dafür, daß die Farben auf dem Drucker durch verschiedene Graustufen dargestellt werden

Flächenfüll-Algorithmus 63
Mit diesem Turbo-Pascal-Programm lassen sich beliebig geformte Figuren auf der Hercules- und IBM-Color-Karte mit einer Farbe ausfüllen

Grafiksymbole beim PC-D 65
Stets „griffbereit" auf dem Ziffernblock der Tastatur macht der Einsatz von Grafikzeichen erst so richtig Spaß

mc-grundlagen

Ein Blick in das Environment 66
Vielleicht wird das Environment unter MS-DOS nur deshalb so wenig eingesetzt, weil man meist auch ohne Kenntnisse darüber auskommt. Vieles jedoch läßt sich damit eleganter und manches überhaupt nur so programmieren – mc gibt einen Einstieg

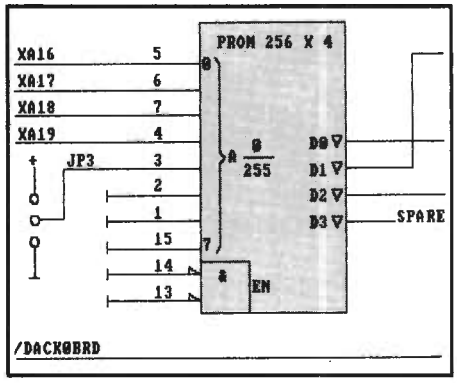
Titelfoto: Klaus Hager

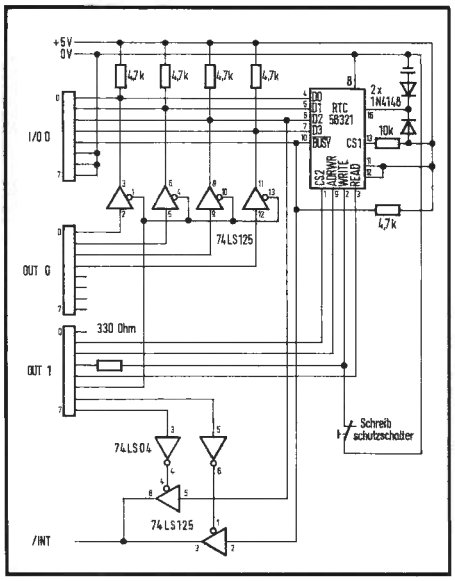
Grafik

Im Mittelpunkt des Interesses stehen seit geraumer Zeit höher auflösende Bildschirm-Darstellungen bei Mikrocomputern. Neben einem Flächenfüll-Algorithmus und einem Programm, das eine Hardcopy von CGA-Farbgrafiken erzeugt, finden Sie einen Beitrag über die 3-D-Darstellung von Funktionen auf dem Atari ST ab **Seite 40**

Tests ...

... um den Mikrocomputer-Markt transparent zu machen. Diesmal haben unsere Tester drei ATs und einen etwas eigenwilligen PC unter die Lupe genommen. Auch ein CAD-Programm für den schmalen Geldbeutel haben wir uns angesehen; unser Urteil lesen Sie ab **Seite 99**





Hardware

Relativ preiswert und sicher für die Besitzer vieler Computer interessant ist der universelle Uhrenbaustein RTC-58321. Seine Hard- und Software-Implementierung im NDR-Klein-Computer sollten sich daher auch Besitzer anderer Rechner ansehen. Sie finden den Beitrag ab

Seite 104

Grundlagen

Wie neu und gut ist denn nun der neue IBM-Mikro-Kanal-Bus? Wir bei mc finden, daß sich ein tieferer Blick in die dahinter stehende Philosophie lohnt – nicht nur für IBM-Fans, sondern auch für alle, die an durchdachten Lösungen Ihre Freude haben. Unsere Serie beginnt auf

Seite 70

Das ist der IBM-Mikro-Kanal-Bus 70
Für Aufsehen in der Fachwelt hat die neue IBM-Rechnerfamilie ja gesorgt – für mc-Leser gibt es gleich handfeste Information: Für's eigene Urteil, für's Know-How und für die Hardware-Entwicklung

Einführung in Unix 78
Im zweiten Teil der Serie geht es nun zur Sache: Mit vielen Beispielen werden eine Reihe von Befehlen gezeigt – so, daß man auch ohne Rechner folgen kann oder, wenn vorhanden, dem Unix-System Schritt für Schritt näher kommt.

mc-AllProg 84
Die unterschiedlichen Programmierverfahren für EPROMs, PROMs und PALs verschiedener Hersteller sind ebenso Inhalt dieses zweiten Teils wie das Protokoll, das zur Übertragung der Daten zwischen AllProg und Steuerrechner verwendet wird

Datenbanksysteme 94
Information – das ist dieser seltsame Rohstoff, den man zum Beispiel in großen, gewerblichen Datenbanken findet. mc zeigt wo und wie

Cursor-Modus beim mc-68000-Computer 95
Ein kleines C-Programm greift direkt auf den 6845 zu – und ist daher auch für MS-DOS-Anwender interessant

mc-test

Drei ATs im Vergleich 96
Wie unterschiedlich sind eigentlich Computer mit der gleichen CPU, fast gleichem Aufbau und auch fast gleicher – fernöstlicher – Herkunft? Dieser Frage ist mc einmal nachgegangen

CAD für den PC 99
Speziell zum Zeichnen von Schaltplänen und Erstellen von Stücklisten scheint uns Shamrock-CAD besonders geeignet – doch auch für andere Anwendungen stellt das getestete Programm eine preiswerte Alternative dar

Spezialitäten-PC 102
Allerlei Besonderheiten heben den Z-159 von Zenith aus der Masse der 8088-PCs heraus – von den Vor- und Nachteilen berichtet mc in diesem Testbericht

mc-hard

Uhrenbaustein RTC-58321 104
Eine Uhr im Rechner ist für vieles nützlich. Und der vorgestellte Baustein hält Hard- und Software-Aufwand wirklich gering – hier bei seinem Einsatz im NDR-Klein-Computer

Impuls-Übertragung via V.24 53
Mit einem kleinen Trick läßt sich eine zusätzliche Schnittstelle einsparen

Siemens PC-D mit NDR-Klein-Bus 98
Ohne Hardware-Änderung läßt sich die Peripherie des NDR-Klein-Computers auch am PC-D nutzen

mc-markt 116

mc-vorschau 150

Impressum 148

```
Procedure DUMP (ENLARGE, INVERSE: Boolean; DENSITY: Byte);
  Begin
    A_REG := 0;
    CALL6502 ($C100);
    MEM [$F779] := 2 + 64 * Byte(ENLARGE) + 32 * Byte(INVERSE);
    MEM [$F479] := DENSITY;
    A_REG := $91;
    CALL6502 ($C102)
  End;
```

Bild 1. Dump und modifiziertes EPROM

Grafik-Dump-Routine

Hier noch etwas Kosmetik für die Grafik-Dump-Routine aus mc 1987, Ausgabe 4 (und Leserbrief in Heft 6 dazu). Die von Herrrn P. Engels vorgeschlagene Änderung der Hardcopy-Routine in Turbo-Pascal ist natürlich nur für das nicht modifizierte Interface-EPROM gültig. Nach der Modifikation wie in mc, Ausgabe 4/87 beschrieben, ist die Option „Doppelte Druckdichte" durch die variable Druckdichte ersetzt. Die Speicherstelle $F479 kann deshalb Werte von 0 bis 7 annehmen; beim Initialisieren wird „5" eingesetzt. Die ursprünglichen Werte, „75" für normale und „76" für doppelte Druckdichte sind jetzt natürlich unsinnig. Die von Herrn Engels vorgeschlagene Dump-Routine könnte für das modifizierte EPROM dann so aussehen, wie es das *Bild 1* zeigt.

Der Aufruf „DUMP (FALSE-,TRUE,5)" zum Beispiel würde dann einen Ausdruck in einfacher Größe, invertiert, bei Druckdichte „5" bewirken. Ob der Input-Buffer unter CP/M gesichert werden muß, das kann nur probiert werden. Ich hatte bisher mit meinem Ehring-Controller keinerlei Schwierigkeiten. Bei dem ohnehin sehr knappen Speicherplatz nach Resrervierung der HGR-Seite, sollten Speicherplatz verbrauchende Operationen nur durchgeführt werden, wenn es nötig ist.

Gerhard Mehrke,
8000 München 80

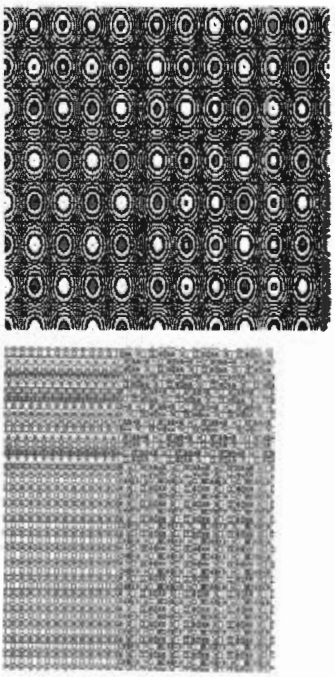


Bild 2. Einige Proben

Cursor ändern unter MS-DOS

In dem Programm in mc 1987, Ausgabe 3 ist angegeben, daß der Cursor wieder normale Größe ereicht, wenn man eingibt:

cursor–normal: mov ch, 12
mov cl, 13

Der Leserbrief gibt hingegen die Werte an:

cursor–normal: mov ch, 6
mov cl, 7

Das Interessante ist, beide haben recht! Wie das?

Des Rätsels Lösung ist, daß die erste Angabe für einen Monochrom-, die zweite aber für einen Color-Bildschirm bestimmt ist. An diesem kleinen Beispiel kann man sehen, wie wichtig es wäre, in solchen Fällen anzugeben, für welchen Bildschirm, beziehungsweise welche Grafik-Karte ein Programm bestimmt ist.

In Bezug auf die Buchbesprechung „Wordstar für PCs und Kompatible" in mc 1987, Ausgabe 5 deckt sich die insgesamt sehr positive Besprechung mit meinen Erfahrungen. Allerdings: Leser, die das Buch wegen Wordstar 3.4 erwerben, werden enttäuscht sein, denn darauf geht das Buch überhaupt nicht ein. Der Autor selbst schreibt in seiner Einleitung: „In diesem Buch wird die Version 3.1 für PCs und Kompatible beschrieben". Alte Hasen wissen, daß damit alle Versionen bis 3.3 eingeschlossen sind, ab 3.4 aber eine vollständig andere Darstellung der Umlaute durchgeführt wird. Offensichtlich ist der Rezensent auf die auffällige Werbung auf der Vorderseite „Vom Junior-Wordstar bis zum Wordstar 3.4 " beziehungsweise auf den letzten Absatz der Inhaltsbeschreibung auf der Rückseite des Buches hereingefallen. Nach Interventionen werden die neueren Bindequoten nun mit dem korrekten Aufdruck" ... bis zum Wordstar 3.3" ausgeliefert.

Dipl.-Ing. Skrzeba,
Studentenwerk Ulm

Druckerumschaltung

Bei dem in mc, Ausgabe 5, 1987 auf Seite 72 vorgeschlagenen Basic-Programm zur Druckerumschaltung sollte in den Zeilen 50 und 110 zur Unterdrückung des Zeilenvorschubs je ein Semikolon ergänzt werden. Der Zeileneditor EDLIN gestattet die Eingabe von ESC durch Ctrl-[, sodaß mit *Bild 1* das Gleiche erreicht wird. Zweckmäßiger ist für die Anwendung allerdings die Erzeugung eines Befehls breit.com mit Debug, wie zum Beispiel in *Bild 2* gezeigt.

Willy Sichtermann,
Münster

```
     1:*^V[W1^Z      'Ctrl-V[W1Ctrl-Z
     2:*^C           'Ctrl-C
*e
```

Bild 1. EDLIN und ESC

```
xxxx:0100 mov cx,03          ' 03  : Anzahl Byte der Sequ.
xxxx:0103 mov bx,111         '111  : Loc. erstes Byte
xxxx:0106 mov dl,[bx]        '       Übergabe der Zeichen
xxxx:0108 mov ah,05          '       Druckerausgabe
xxxx:010A int 21
xxxx:010C inc bx
xxxx:010D loop 106
xxxx:010F int 20
xxxx:0111 db 1b 'W1'         '     : ESC-Sequenz breit
```

Bild 2. Breit.com mit DEBUG

PC-Grafik-Symbole

Die Idee auf Seite 116 in mc 5 1987, die für das Zeichnen von Rahmen erforderlichen Grafik-Symbole einmal übersichtlich darzustellen ist sehr gut. Nur fehlen leider in Ihrer Veröffentlichung einige Symbole – und so ideal zum Heraussuchen eines gerade benötigten Symbols ist die Tabelle auch nicht. Ich schlage daher die im *Bild* gezeigte Anordnung als eine Alternative vor. *Detlef Pirk,*
1000 Berlin

Kreuze	┼197	╪216	╫215	╬206
Ecken	218┌ ┐191	213╒ ╕184	214╓ ╖183	201╔ ╗187
	192└ ┘217	212╘ ╛190	211╙ ╜189	200╚ ╝188
T-Stücke	195├ ┤194	198╞ ╡209	199╟ ╢210	204╠ ╣203
	193┴ ┬180	207╧ ╤181	208╨ ╥182	202╩ ╦185
Geraden	179│ ─196			205═ ║186

Bild. Die Grafik-Symbole schnell gefunden

Bilder digitalisieren, verwalten und mit dem Laserdrucker wiedergeben: Das Document-Image-Processing-System macht es möglich

PC verwaltet Dokumente

Document-Image-Processing nennt sich ein Produkt, bestehend aus Hardware und Software, mit dem man Dokumente digitalisieren, am hochauflösenden Schwarzweiß-Monitor weiterverarbeiten und auf dem Laserdrucker ausgeben kann. Den Alleinvertrieb dafür hat in der Bundesrepublik Deutschland die Firma MikroGraf.

Mit dem System lassen sich Vorlagen wie Krankenscheine, Banküberweisungen oder andere Belege digital „fotokopieren" und in das Archiv übernehmen, das der PC verwaltet. Sind die Vorlagen erst einmal in dieser Form gespeichert, dann kann man sie natürlich auch elektronisch übertragen. Dies bietet z. B. Banken die Möglichkeit, die Unterschrift von Kunden zwischen den einzelnen Filialen auszutauschen und zu vergleichen.

Das Document-Image-Processing-Paket kostet deutlich unter 50 000 DM. Es ist in der Lage, pro Stunde etwa 240 DIN-A4-Seiten zu verarbeiten. Auf einer 20-MByte-Harddisk bringt man rund 150 DIN-A4-Seiten unter. Bei höherem Speicherbedarf kann man auch optische Speicher anschließen.

Desktop Publishing: Präsentationsgrafik ergänzt GEM-Software

Digital Research ergänzt jetzt seine Programme der GEM-Familie um ein Software-Paket für die Gestaltung von Druckvorlagen. Der „GEM-Desktop-Publisher" ist in der Lage, Text und grafische Darstellungen zu mischen. Dabei weist er eine wichtige Eigenschaft auf, die die Amerikaner mit WYSIWYG abkürzen, was soviel bedeutet wie: „What you see is what you get." Das heißt nichts anderes, als daß auf dem Bildschirm genau das zu sehen ist, was später der Drucker zu Papier bringt. Grundlage des Programms ist die grafische Benutzerschnittstelle GEM, die auf Bildsymbolen basiert. Dies macht den Umgang mit den Programmen so einfach, daß der Anwender nach kurzer Zeit produktiv arbeiten kann.

Der GEM-Desktop-Publisher verarbeitet Textdateien, die mit Programmen wie GEM-Write, WordPerfect, WordStar, MultiMate oder DisplayWrite-3 erstellt worden sind. Mit Hilfe eines besonderen Tastaturanpassungsprogramms läßt sich das Programm mit den gleichen Steuerbefehlen bedienen, die man von seinem Texteditor her kennt. Lauffähig ist der GEM-Desktop-Publisher auf IBM-PCs und auf der neuen Familie System/2 von IBM. Die deutsche Version wird ab Mitte Juli zu einem Preis von knapp 1400 DM erhältlich sein.

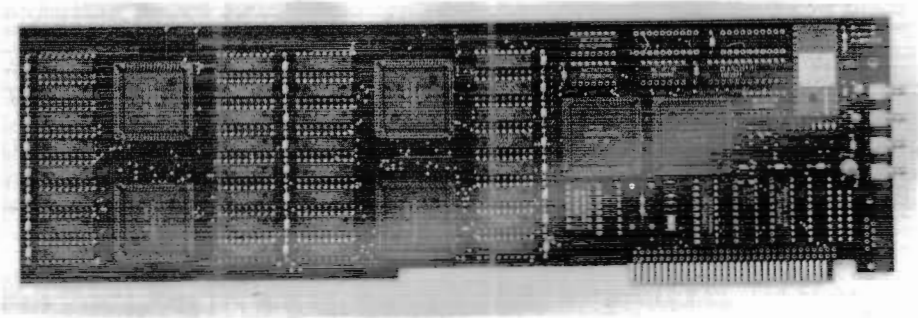
Neues Sinix-Modell ist grafikfähig

Mit dem neuen Modell PC-X10, das auf dem Unix-kompatiblen Betriebssystem Sinix basiert, hat Siemens einen grafikfähigen Personal Computer auf den Markt gebracht. Wie sein Vorgänger PC-X ist auch dieses Gerät kompatibel zu allen anderen Mitgliedern der Sinix-Familie. Seine besondere Stärke liegt in der einfachen und komfortablen Bedienung mit der Maus. Die Benutzeroberfläche „Collage" ermöglicht die Arbeit mit mehreren gleichzeitig geöffneten Fenstern. Zeitraubende Programmwechsel entfallen dabei in vielen Anwendungen.

Das System enthält die genormte Grafikschnittstelle GKS. Auf seinem 12-Zoll-Bildschirm lassen sich Zeichnungen mit 640 x 400 Bildpunkten (monochrom) darstellen.

Der Hauptspeicher des Rechners hat eine Kapazität von 0,5 oder 1 MByte. Disketten-Laufwerke stehen mit 1 oder 1,6 MByte zur Verfügung, an Festplatten hat der Kunde die Wahl zwischen Kapazitäten von 12,8 oder 25,6 MByte. Zahlreiche Kommunikationsschnittstellen ermöglichen den Anschluß an verschiedene Computernetze oder öffentliche Dienste wie Telex und Btx. Besitzer des Vorgängermodells können mit einem Grafik-Umrüstsatz ihr altes System auf den neuesten Stand bringen.

Als Einzelblattsystem oder als Arbeitsstation im Netzwerk läßt sich der PC-X10 von Siemens betreiben



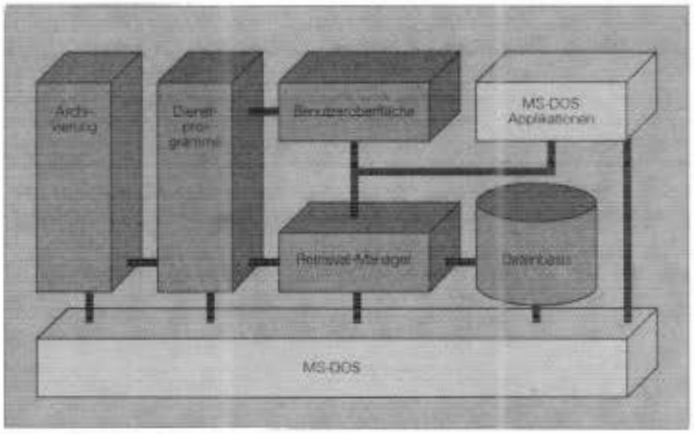
Für CAD/CAM-Anwendungen eignet sich diese PC-Grafikkarte

PC wird zur Workstation

Als Workstation bezeichnet man sehr leistungsfähige Tischcomputer, die sich wegen ihrer grafischen Möglichkeiten insbesondere im CAD/CAM-Bereich durchgesetzt haben. Die üblichen IBM-kompatiblen PCs gehören normalerweise nicht zu dieser Gerätekategorie. Erweitert man einen PC allerdings um eine Grafikkarte, wie sie die Firma Pro Didaktik mit dem Modell GRA-3 anbietet, dann kann man getrost von einer Workstation sprechen.
Diese Zusatzkarte enthält Hitachis Grafikcontroller HD63484 (ACRTC), dessen Eigenschaften voll genutzt werden. Sie kann im Bildspeicher bis zu 2048 × 2048 Bildpunkte zu je vier Bit unterbringen. Die Karte verfügt über 16 Farbpaletten, die eine Darstellung von jeweils 16 Farben aus insgesamt 262 144 ermöglichen. Jede dieser Farbpaletten ist verschiedenen Bereichen des Bildschirms zugeordnet. Insgesamt lassen sich so auf dem Bildschirm maximal 128 Farben gleichzeitig darstellen.
Mit den verschiedenen Versionen dieser Karte erreicht man einen Punkttakt von 40, 48 oder 64 MHz. Die maximale Auflösung beträgt 1024 × 768 Bildpunkte bei einer Bildwiederholrate von 60 Hz (ohne Zeilensprungverfahren).

Daten auf CD-ROMs aufbewahren und wiederfinden

Optische Speichermedien, die unter Namen wie CD-ROM oder WORM-Laser-Disc bekannt sind, bieten seit kurzem auch PC-Benutzern die Möglichkeit, riesige Datenmengen zu archivieren. Um sich im Wust dieser Daten wieder zurechtzufinden, ist spezielle Software erforderlich. Ein sogenanntes „Information Retrieval System" für MS-DOS- und Unix-Rechner, das diese Aufgabe erledigt, vertreibt jetzt die Firma Systec, Berlin. In Verbindung mit CD-ROM-Laufwerken ist das Programmpaket GigaBase in der Lage, bis zu 250 000 DIN-A4-Seiten zu speichern.
Der Zugriff auf eine beliebige Stelle dauert etwa ein bis drei Sekunden. Eine besondere Form muß man beim Abspeichern von Texten oder Zeichnungen nicht einhalten. So lassen sich Zeitschriftenartikel, Statistiken, technische Zeichnungen und ähnliches völlig formatfrei überspielen. Auch der Telex- oder Btx-Verkehr kann ohne besondere Aufbereitungsmaßnahmen direkt archiviert werden. Durchsucht werden Texte einfach nach beliebigen Begriffen, nicht etwa nach Schlüsselwörtern wie bei üblichen Datenbanken.



So stellt sich die Implementation von GigaBase auf einem MS-DOS-Rechner dar

TI entwickelt Lisp-Chip

Funktion und Einsatzmöglichkeiten de ersten 32-Bit-Mikroprozessor-Chips, de speziell für Anwendungen im Bereich de künstlichen Intelligenz entwickelt wurde standen im Mittelpunkt der Studie eine führenden KI-Experten von Texas Instru ments, die erstmals auf der letzten Inter national Solid-State Circuits Conferenc der Öffentlichkeit präsentiert wurde. I seinem Arbeitspapier mit dem Titel „ 553K-Transistor Lisp Processor Chip" er läuterte Patrick W. Bosshart, leitende Mitarbeiter im Semiconductor Proces and Design Center von TI, Möglichkeite und Eigenschaften dieses neu entwickel ten Bausteins.
Lisp ist (neben Prolog) eine der beide Sprachen, die das Gebiet der künstliche Intelligenz dominieren. Herkömmlich Prozessoren sind dafür nicht optimiert. Ir Vergleich dazu führt der neue 32-Bit-Chi von Texas Instruments Lisp-Programm wesentlich effektiver aus. Damit hat ma jetzt auch bei KI-Anwendungen die Mög lichkeit, leistungsfähige Systeme m kompakter Hardware zu entwickeln.
Mit über 550 000 Transistorfunktionen au einem Quadratzentimeter gehört der Bau stein heute zu den Halbleitern mit de höchsten Integrationsdichte. Entwickel wurde er im Rahmen eines Auftrags de US-Regierung. Derzeit ist eine Version i Vorbereitung, die auch in die Massenpro duktion gehen soll.

Mehr Leistung, geringerer Preis

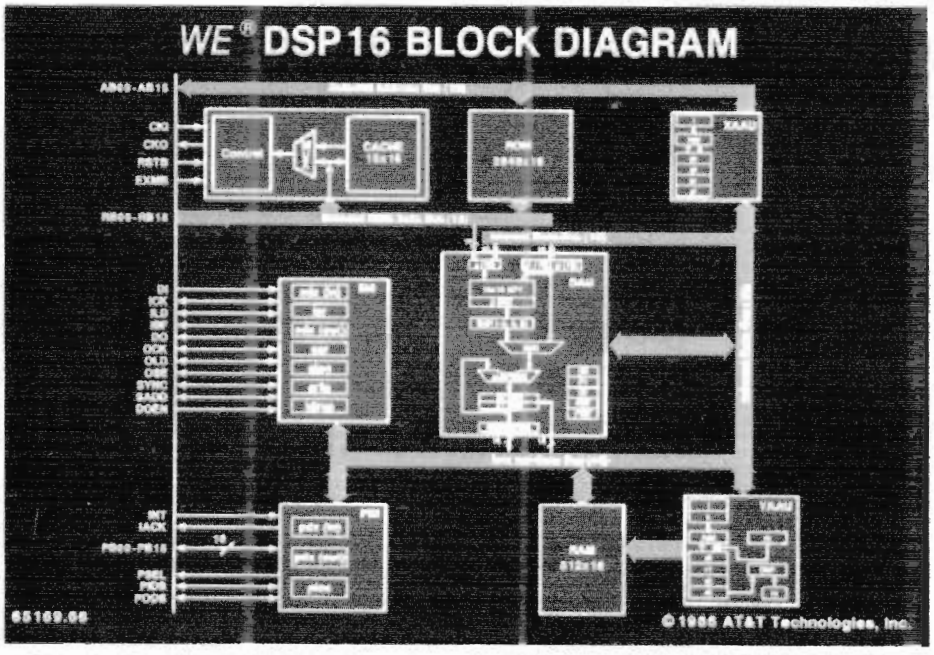
Alle Modelle der AT-kompatiblen PCA Serie von Tandon sind ab sofort mit MByte Hauptspeicher (bisher 512 KByte) Hercules-kompatibler Monochrom-Gra fikkarte, serieller und paralleler Schnitt stelle sowie mit dem Programm MS-Win dows ausgestattet. Auch die XT-kompati blen Modelle der XPC-Serie enthalte neuerdings die Hercules-Grafik. Gleich zeitig hat das Unternehmen neue Preis angekündigt. So kostet beispielsweis das leistungsstärkste Modell, PCA 70 (A mit 70 MByte Festplatte) künftig nur nocl 9995 statt bisher 10 495 DM. Noch stär ker als bei den PCA-Modellen gingen di Preise bei der XPC-Serie zurück. Da preiswerteste Gerät, ein XT mit zwe Laufwerken, kostet jetzt 2495 DM (bishe 3995 DM).

Preiswertes PC-Netz

Mit der neuesten Version des Low-Cost-Netzwerks NET.24 bietet die Münchener Shamrock Software GmbH PC- und AT-Benutzern die Möglichkeit der hausinternen Bürokommunikation zu geringen Kosten (98 DM pro Station). Eine Teilnehmer-Ausstattung besteht neben dem (deutschen) Handbuch aus einer Diskette für IBM-kompatible PCs bzw. ATs sowie einem kleinen Interface, das direkt in die serielle Schnittstelle des Computers gesteckt wird und den Datentransfer zwischen zwölf Netzwerk-Teilnehmern über eine einfache abgeschirmte Leitung (Koax) ermöglicht. Die Länge dieser als Bus dienenden Leitung kann 50 m betragen. Der Datentransfer erfolgt mit 4800 Bit/s.
Die speicherresident ladbare NET.24-Software kann über eine Tastenkombination jederzeit aufgerufen werden, um mit anderen Netzwerk-Teilnehmern in Dialog zu treten oder Textdateien oder Programme zu transferieren. Aber auch dann, wenn ein Teilnehmer gerade in einem Anwenderprogramm arbeitet, ist er nicht unerreichbar: Man kann ihn jederzeit mit einer Adresse (A...Z) rufen, so daß er einen kurzen Signalton hört, während im rechten oberen Eck seines Bildschirms etwa 14 Sekunden lang ein blinkendes „N" erscheint, um ihn aufmerksam zu machen, das Netzwerk-Programm einzublenden. Mit Adresse A kann man auch alle Teilnehmer gleichzeitig rufen.
Damit bietet NET.24 im Gegensatz zu anderen Netzwerken (die in erster Linie ein Disketten-Laufwerk außerhalb des eigenen Rechners simulieren) eine leicht installierbare und sehr preiswerte Möglichkeit zur Bürokommunikation innerhalb eines Betriebes. Darüber hinaus ist es aber auch möglich, einen unbedienten PC als File- und Drucker-Server einzusetzen; die dafür nötige Zusatz-Software wird nur einmal benötigt und ist ebenfalls für 98 DM zu haben.

Schneller Signalprozessor

Einen 16-Bit-Signalprozessor mit Festkomma-Arithmetik hat die Firma AT&T entwickelt. Der WE-DSP16 braucht für eine Multiplikation mit anschließender Addition nur 55 ns. Damit ist er nach Überzeugung des Herstellers der schnellste Einchip-Signalprozessor mit Festkomma-Arithmetik, der derzeit angeboten wird.
Eine Besonderheit des neuen Bausteins ist der 15 Worte große Cache-Speicher, den man mit Anwenderprogrammen laden kann. Da die Algorithmen der digitalen Signalverarbeitung in der Regel sehr einfach, aber häufig auszuführen sind, reicht für die meisten Anwendungen eine Speichertiefe von 15 Worten völlig aus. Hergestellt wird der WE-DSP16 in CMOS-Technik mit Strukturbreiten von 1 µm. In die Serienfertigung soll er im dritten Quartal dieses Jahres gehen.



In 55 ns führt AT&Ts neuer Signalprozessor eine Multiplikation mit Addition aus

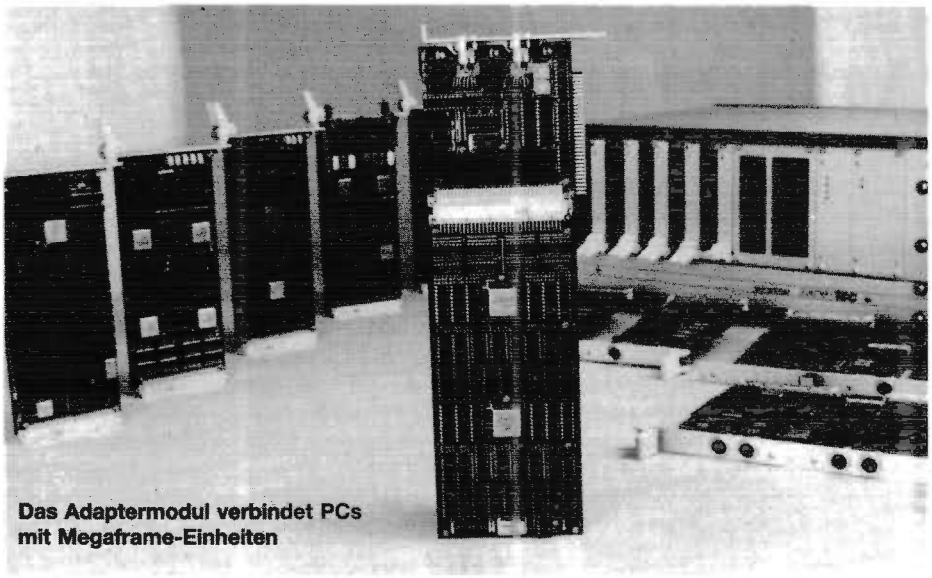
Ein Entwicklungssystem für die 68000-Familie hat die Firma Instrumatic vorgestellt

68000-Entwicklungen auf dem PC durchgeführt

Ein Entwicklungssystem für die Prozessoren der 68000-Fammilie hat die Firma Instrumatic vorgestellt. Es besteht aus Makroassembler, C-Cross-Compiler, C-Hochspracher-Debugger und In-Circuit-Emulator (von AMC). Der Debugger läuft auf dem IBM-PC und bedient sich der Fenstertechnik. Er erlaubt unter anderem den Einzelschrittbetrieb auf Assembler- oder C-Ebene. Außerdem ist von diesem Dienstprogamm aus der Zugriff auf den Satelliten-Emulator ES-1800 möglich, der neben den Prozessoren der 68000-Familie auch Intels 8086, 80186 und 80286 sowie Zilogs Z8000-Familie unterstützt. Dieser Emulator ist mit 2 MByte Overlay-RAM und 2 KByte Echtzeit-Trace-RAM bestückt. Ein komplexes Triggersystem erhöht den Komfort des Emulators.

Tandon zieht dicken Fisch an Land

Einen wahrlich dicken Fisch hat der kalifornische Laufwerk- und PC-Hersteller Tandon an Land gezogen: Das Unternehmen erhielt einen Auftrag zur Lieferung von 3,5-Zoll-Festplatten-Laufwerken im Wert von rund 50 Mio. $. Auftraggeber ist der britische PC-Anbieter Amstrad. Allein im ersten Halbjahr dieses Jahres wird der Wert der ausgelieferten Produkte bei 30 Mio. $ liegen.



Das Adaptermodul verbindet PCs mit Megaframe-Einheiten

PC nutzt Transputer

Das Transputer-Konzept der britischen Firma Inmos basiert darauf, daß viele CPUs gitterartig miteinander verbunden sind und ihre Arbeit parallel erledigen. Damit kommt man zu superschnellen Recheneinheiten. Einheiten dieser Art bietet z. B. die Firma Parsytec unter der Bezeichnung Megaframe an. Jetzt hat die Firma ein Adaptermodul für 980 DM vorgestellt, mit dem man die Komponenten der Megaframe-Familie auch über IBM-PCs betreiben kann.
Das Adaptermodul, das in den PC-Slot eingesteckt wird, ist für zwei Verwendungsarten vorgesehen. Zum einen nimmt es über eine Steckverbindung ein beliebiges Megaframe-Modul auf. In dieser Betriebsart stellt die Transputer-Einheit einfach eine Erweiterung des PCs dar. Die erforderlichen MS-DOS-Treiber sind vorhanden.
Ebenso kann das Adaptermodul aber auch allein in den PC-Slot eingesteckt werden. Über ein Standard-Link-Kabel läßt sich dann eine Verbindung mit anderen Transputer-Systemen oder über „Bus-Brückenköpfe" zu VME-, SMP- oder ECB-Systemen herstellen. Das Kabel kann bis zu 10 m lang sein; die Übertragungsgeschwindigkeit beträgt 20 MBit/s.

Zorland-C-Compiler in verbesserter Version

Der Zorland-C-Compiler, der im deutschen Sprachraum von CCP-Software angeboten wird, liegt nun in der Version 2.0 vor. Gegenüber der ersten Version wurde der Leistungsumfang erheblich gesteigert. So wurden z. B. zahlreiche neue Bibliotheksfunktionen (u. a. eine komplette Routinensammlung zur Unterstützung der Microsoft-Maus) implementiert. Der Lieferumfang umfaßt nun neben dem Compiler selbst und dem Quellcode aller Bibliotheksfunktionen einen neuen Editor, den Zorland-Speed-Linker und den Zorland-Librarian.
Der Editor gestattet das gleichzeitige Bearbeiten von maximal sechs Programmen, die Definition von Tastaturmakros und das direkte Compilieren der editierten Programme, wobei er nach einer Fehlermeldung des Compilers sofort in die fehlerhafte Programmzeile springt. Zum Compiler werden zwei deutschsprachige Handbücher – ein Referenzhandbuch und ein Tutorial – mit insgesamt 450 Seiten mitgeliefert. Neu eingerichtet wurde auch die Zorland-Hotline, die an drei Tagen in der Woche jedem registrierten Zorland-Benutzer kostenlos zur Verfügung steht. Der Preis von 259 DM für das Komplettpaket ist gleich geblieben.
Gleichzeitig mit der Vorstellung der neuen Compiler-Version wurden die ersten beiden Zorland Toolboxen präsentiert. Die Graphics-Toolbox ermöglicht die Erstellung hardwareunabhängiger Grafikprogramme. Sie unterstützt nicht nur zahlreiche Grafikkarten und hochauflösende Grafiksubsysteme, sondern ermöglicht auch die Ansteuerung nahezu aller marktgängigen Drucker. Mit über 60 Grafikroutinen steht mit der Graphics-Toolbos ein leistungsfähiges Werkzeug zur Verfügung.
Die Data-Toolbox ist eine Sammlung von Dateiverwaltungsroutinen, die die Programmierung von Datenbanken nach der ISAM- oder BTREE-Methode unterstützt. Darüber hinaus können auch Dateien mit variabler Datensatzlänge verwaltet werden. Für beide Programmpakete stehen Demo-Disketten zur Verfügung, die für 15 DM von CCP Software erhältlich sind. Im Laufe dieses Jahres sind noch vier weitere Toolboxen – darunter ein globaler Optimizer und eine Window-Toolbox – geplant. Jedes dieser Programmpakete wird 198 DM kosten.

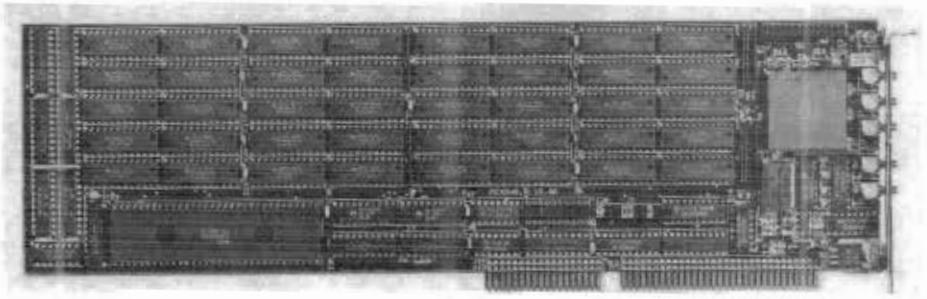
Olivetti legt wieder zu

Nachdem der Umsatz im vergangenen Jahr leicht zurückgegangen war, gab es im ersten Quartal für die deutsche Olivetti GmbH wieder eine Steigerung von 5 % gegenüber dem gleichen Vorjahreszeitraum. Die Personal-Computer-Stückzahlen gingen sogar um 28 % nach oben. Damit konnte man aber gerade den Preisrückgang in dieser Sparte ausgleichen.
Das Geschäftsjahr 1986 schloß die deutsche Olivetti GmbH mit einem Umsatz von 453,5 Mio. DM (1985: 469,2 Mio. DM) ab. Davon entfielen 64,4 % auf die Hardware, 27,1 % auf Software und Dienstleistungen sowie 6,2 % auf Zubehör. An PCs wurden insgesamt 20 500 verkauft. Dies entspricht auch der Stückzahl des Jahres 1985.

Turbo-Karte mit Nortonfaktor 10,1

Diesen beeindruckenden Wert erreicht eine Beschleunigerkarte für PCs, mit der man das Modell XT zum XT/286 aufrüsten kann. Das Produkt, das die Bezeichnung Breakthru286 trägt, gab es bisher in einer 8-MHz-Version, jetzt ist es mit 12 MHz Taktfrequenz erhältlich. Aber das allein reicht noch nicht aus, um die zehnfache Geschwindigkeit gegenüber dem Standard-PC zu gewährleisten. Zusätzlich sorgen ein Arithmetikprozessor (80287) und ein 16-KByte-Cache-Speicher für höchste Geschwindigkeit.
Der Cache-Speicher verkürzt die Zeit für Plattenzugriffe – im Schnitt zwischen Faktor 2 und 4. Hersteller der Karte ist die amerikanische Firma PCSG, vertrieben wird sie von der Wiener Firma P.C.S.

Grafik für Profis



Professionellen Ansprüchen genügt diese Grafikkarte für den IBM-PC

Die Grafikkarte mit dem etwas unhandlichen Namen PC-XXHR der Aachener Firma Elsa ist eine Weiterentwicklung der PC-XHR. Sie eignet sich für IBM-PC-kompatible Computer und basiert auf dem Grafikprozessor HD63484 von Hitachi. Ihr Video-Punkttakt beträgt bis zu 110 MHz. Mit 1,3 MByte Bildspeicher und einer CMOS-Farbpalette lassen sich aus 4096 Farben 256 darstellen. Die Auflösung beträgt dabei 1280 × 1024 Bildpunkte. In einer Sonderversion stehen sogar 16 Mio. Farben zur Verfügung.
Die Karte gibt es in zwei Ausführungen: Die eine arbeitet mit einem Punkttakt von 75 MHz und stellt 1024 × 786 Bildpunkte dar. Dafür ist ein Monitor mit 50 kHz Zeilenfrequenz erforderlich. Die zweite Ausführung mit 110 MHz Punkttakt stellt die erwähnten 1280 × 1024 Punkte dar und erfordert einen Monitor mit 64 kHz Zeilenfrequenz.
Der auf der Karte vorhandene Grafikprozessor entlastet die CPU von allen zeitaufwendigen Zeichenoperationen. Punkt, Linie, Kreis, Ellipse, Kreis- und Ellipsensegmente sowie Füll- und Musteroperationen sind hardwaremäßig eingebaute Funktionen dieses Bausteins. Die komplette Bildspeicherverwaltung übernimmt der Grafikprozessor ebenfalls.
Die meisten Programme, die an die Vorgängerversion der Karte angepaßt wurden, laufen auch auf dem neuen Produkt, z. B. AutoCAD, VersaCAD, MikroGraf-MG86, Lotus-1-2-3, CADdy, INTERCAD, PC-Draft und viele andere mehr.

Erfolg mit Transputern

Einen beachtlichen Erfolg konnte die junge Aachener Firma Parsytec verbuchen. Auf Einladung des japanischen Konzern Matsushita stellte das Unternehmen Mitte April ihr Megaframe-System eine Woche lang dem japanischen Fachpublikum auf der Tokyoter „Technology Japan 87" vor. Die Megaframe-Parallelrechner bauen auf den Transputer-Chips der Firma Inmos auf. Damit lassen sich praktisch beliebig viele Prozessoren zu einem Netz zusammenschließen. Rechner dieser Art dringen immer mehr in Bereiche wie Robotersteuerungen, Bildverarbeitung und künstliche Intelligenz vor.

Nach der Ausstellung, auf der die Megaframe-Familie großes Interesse hervorgerufen hatte, sicherte sich Matsushita die exklusiven Vertriebsrechte für den japanischen Markt. Das erste Megaframe-System wurde bereits vom Forschungsinstitut der japanischen Atomenergiebehörde in Auftrag gegeben.

Großes Interesse riefen die auf den Transputer-Chips basierenden Megaframe-Produkte der Aachener Firma Parsytec in Japan hervor

Modula-2 für IBM-PC und Amiga

Die schweizerische Firma A.+L. Meier-Vogt bietet jetzt den Modula-2-Compiler des Software-Entwicklungssystems M2SDS (für den IBM-PC) auch separat an. Er kann zusammen mit jedem Editor und Linker verwendet werden. Damit der Benutzer nicht ganz auf dem Komfort der Window- und Bibliotheksumgebung verzichten muß, liefert die Firma kostenlos eine Make-Funktion mit. Der Compiler führt den zweiten Durchlauf nur dann aus, wenn dies nötig ist. Dadurch wird er extrem schnell. Er erzeugt Maschinencode, der allerdings im Gegensatz zu dem der integrierten Version um 10...20 % langsamer ist. Dafür gibt es auch mit solchen PCs keine Schwierigkeiten, die nicht voll IBM-kompatibel sind. Preis: 300 DM.
Für denselben Betrag bietet die Firma einen Modula-2-Compiler an, der auf dem Amiga läuft. Nach Angaben des Unternehmens handelt es sich dabei um den schnellsten Modula-2-Compiler für diese Maschine – in bezug auf Compilier- und auf Ausführungszeiten. Er ist vollständig in das Betriebssystem des Amiga eingebettet und kann auch alle Amiga-Funktionen ansprechen. In diesem Software-Entwicklungssystem sind außerdem enthalten: ein Editor, der Fehler im Klartext anzeigt, ein schneller Linker, ein Laufzeitsystem, das insbesondere das Multitasking-System des Amiga berücksichtigt, sowie zahlreiche Grundbibliotheken mit den Schnittstellen zum Betriebssystem.

Die schnellsten
Modula-2
Software-Entwicklungssysteme
für

AMIGA™ DM 300.– +MWSt.
SFr. 270.–

Single-Pass-Compiler, extrem schnell, in Workbench integriert, volle Unterstützung aller dokumentierter Amiga-Funktionen (Intuition, Exec, usw.), Typen doppelter Genauigkeit und FFP, erzeugt schnellsten Maschinencode, linkt in wenigen Sekunden!
Inkl. Editor, Compiler, Linker, Module, deutsche Beschreibung und englischem Einführungsbuch.
Minimalkonfiguration: 512 kByte, 1 Laufwerk.

Zuschlag für zusätzliches deutsches Einführungsbuch DM 35.–/SFr. 30.–
Demodiskette SFr./DM 10.–

IBM/370-Mainframes
SFr. 16000.–

Single-Pass-Compiler, extrem schnell (bis zu 36000 Zeilen pro Minute), volle 32-Bit-Arithmetik, getrennte Übersetzung mit allen Vorteilen von Modula-2 (Versionskontrolle, Kompatiblitätsprüfung, Typechecking über die Modulgrenzen hinweg!), Schnittstellen zu Assembler und Fortran, Unterstützung von Projektbibliotheken, erzeugt schnellsten Native-Code (mit Arithmetik-Check) für MVS-Linker-Loader.

Jährlicher Wartungsvertrag SFr. 2750.–

IBM PC/XT/AT
DM 299.90 +MWSt.
SFr. 267.50

Inkrementeller oder Zwei-Pass-Compiler, extrem schnell, Window-System oder konventionell, mit und ohne syntaxgesteuertem Editor, viele, viele Zusatzfunktionen, erzeugt schnellsten Maschinencode.

Wir haben ein umfassendes Angebot von Toolboxen und Werkzeugen für Modula-2, verlangen Sie unsere Unterlagen und Demo-Disketten. Erkundigen Sie sich auch nach unseren interessanten
Public-Domain-Disketten
zu Modula-2

Bezugsquellen:
Bundesrepublik Deutschland:
– Interplan, Haslacher Weg 95, 7900 Ulm, 0731/2 69 32
– E. Jurschitza, Ellensindstr. 7a, 8900 Augsburg, 0821/8 57 37
– SW-Datentechnik, Raiffeisenstr. 4, 2085 Quickborn, 04106/39 98
– Wilken & Sabelberg, Kasernenstr. 26, 3300 Braunschweig, 0531/33 21 17
Schweiz:
– Frei-Elektronik, Stationsstr. 37, 8604 Volketswil, 01/945 54 32
Österreich:
– ICA GmbH, Heigerleinstr. 9, 1160 Wien, 0222/45 45 010

oder bei Ihrem nächsten Computer- oder Buchhändler

BD **Generalvertrieb für Europa:**
A. + L. Meier-Vogt
Im Späten 23
CH-8906 Bonstetten/ZH
Tel. (41) (1) 700 30 37
E-Mail: APLUSL@komsys.ifi.ethz.chunet (UUCP)

ERF wir sprechen von Gott
Täglich im Radio.
Zu hören in ganz Europa.
Hören Sie doch mal rein!
5^45+21^30 h MW 1467 kHz
10^00+12^00+15^30 h KW 9230 kHz 49 m Band
KW 7205 kHz 41 m Band
Gerne schicken wir Ihnen weitere Informationen.
Name
Adresse
Senden Sie den Coupon bitte an: ERF, D-6330 Wetzlar.

„Schulis" soll Computer-Chaos an der Schule beenden

An den bundesdeutschen Schulen herrscht, was den Computer betrifft, ein Chaos. Vielfach steht und fällt der gesamte Computerunterricht mit einem einzigen Lehrer, in der Regel ein Idealist, der sich als Einzelkämpfer betätigen muß. Qualitativ hochwertige Unterrichts-Software gibt es kaum, und die Kultusminister der Bundesländer können sich nicht dazu entschließen, einheitliche Richtlinien für den Einsatz von Computern im Unterricht zu erlassen. So stehen in unseren Schulen die verschiedensten Computer, auf denen die verschiedensten Betriebssysteme und Programmiersprachen laufen.
Die Gesellschaft für Mathematik und Datenverarbeitung (GMD), die für Informatik und Informationstechnik zuständige Großforschungseinrichtung der Bundesrepublik Deutschland, will jetzt diese Misere beenden. Sie hat mit Schulis ein komplettes Software-Paket für die Schule entwickelt, von dem jetzt die ersten Teile zum Einsatz bereitstehen. Schulis deckt die drei wichtigsten Bereiche des Rechnereinsatzes an Schulen ab:
– Computer als Lerngegenstand im Informatik-Unterricht,
– Computer als Medium und Werkzeug im Fachunterricht,
– Computer als Unterstützungsmedium für die Schulverwaltung.

Das Programmpaket basiert auf dem Betriebssystem Eumel und der Programmiersprache Elan.
Eumel ist ein von der GMD in Zusammenarbeit mit der Universität Bielefeld speziell für Mikrocomputer entwickeltes, erweiterbares Betriebssystem für Mehrbenutzerbetrieb auf der Basis von Elan. Sein wichtigster Vorteil: Es kann ohne große Schwierigkeiten von einem Computer-Fabrikat auf ein anderes übertragen werden. Fast 40 unterschiedliche Computersysteme können bereits mit Eumel arbeiten. Anpassungen an neue Rechnergenerationen sind in Vorbereitung. Darüber hinaus sind die GMD-Wissenschaftler dabei, Eumel an MS-DOS anzupassen.
Elan ist eine vom Arbeitskreis Schulsprachen für den Unterricht empfohlene Programmiersprache, die aber in den Bundesländern noch zu wenig bekannt ist. Sie ist von Laien leicht erlernbar und erlaubt den Gebrauch der natürlichen Sprache.
Für den Informatikunterricht der Sekundarstufen I und II bietet Schulis einen professionellen „Werkzeugkasten" mit einer Fülle geeigneter Hilfsmittel. Die Palette reicht von Hilfen zum Erlernen der Programmiersprache bis hin zu ausgereiften Werkzeugen. Da das Betriebssystem selbst Gegenstand des Unterrichts ist, wird sogar der Quellcode von Eumel offengelegt.
Als didaktische Hilfe für den Einstieg in den Informatikunterricht wurde das „Hamster"-Modell entwickelt, das, beginnend mit nur vier Grund- und drei Testbefehlen, dem Anfänger schon nach kurzer Zeit erlaubt, sich selbst kleine Aufgaben zu stellen und zu lösen. Darüber hinaus steht mit dem Schulis-Modellcomputer ein System zur Verfügung, mit dessen Hilfe die Prinzipien der Computertechnik veranschaulicht werden können.

Computer als Unterrichtsmedium

Im Fachunterricht steht nicht das Programmieren, sondern die Anwendung fertiger Programme im Vordergrund. Hier kommt dem Computer die Rolle eines zusätzichen Unterrichtsmediums zu. Sein Einsatz ist nur dort sinnvoll, wo er eine Verbesserung und Erleichterung des Unterrichts ermöglicht oder Inhalte erschließen hilft, die mit herkömmlichen Hilfsmitteln nicht oder nur schwer zu vermitteln sind. Nach diesen Kriterien werden vor allem vier Einsatzformen als wichtige Bereicherungen des Unterrichts angesehen:
– Mit Hilfe von Simulationsprogrammen können formalisierte Modelle genutzt werden, um bestimmte Zusammenhänge beispielsweise in der Biologie, der Chemie, der Physik, den Sozialwissenschaften und der Geographie zu verdeutlichen.
– Informationssysteme können Fachwissen und Datenbestände für Lehrer und Schüler bereitstellen, mit deren Hilfe sie eigene Fragestellungen entwickeln und bearbeiten können.
– Bei Trickfilmen und „elektronischen Wandtafeln" werden die grafischen Möglichkeiten des Computers genutzt, um beispielsweise funktionale Zusammenhänge oder zeitliche Dynamik zu veranschaulichen.
– Im naturwissenschaftlichen Unterricht kann der Computer Experimente automatisch steuern oder Meßwerte registrieren und auswerten, wenn entsprechende Geräteanschlüsse vorhanden sind.

Fachunterrichtsprogramme sind für die Bereiche Biologie, Chemie, Mathematik, Physik und Gesellschaftswissenschaft vorgesehen. Hierfür ist das Schulis-Simulationssystem Simsel gedacht, ein für Laien entwickeltes Simulationswerkzeug mit verschiedenen Modellen. Lehrer und

mc-info

Schüler können mit Simsel ökologische, biologische oder chemische, aber auch gesellschaftliche Systeme studieren, indem sie Ausgangswerte und Bedingungen variieren und die dadurch ausgelösten Veränderungen beobachten und interpretieren.
Auch für die Schulverwaltung bietet Schulis ein komplettes System, mit dem Sekretärinnen, Schulleiter und Lehrer problemlos und ohne Informatik-Vorkenntnisse umgehen können.
Als erste Komponente ist das Teilsystem Schülerdaten-Verarbeitung einsatzbereit. Stichworte dazu: Klassen- und Kurslisten für jeden Klassen- und Fachlehrer, Klassenbuchlisten zum Einkleben in die Klassenbücher, Schülergesamtlisten für das Sekretariat und Bescheinigungen des Schulbesuchs für alle Schüler.
Die weiteren Komponenten werden Unterstützung zu Kurswahlen und Abiturbegleitung, Stundenplanung und Stundenplanauswertung, Erstellen der amtlichen Statistiken und Inventarverwaltung bieten. Mit Hilfe eines speziellen Programms wird es dem Anwender möglich sein, sich eigene Listen und statistische Auswertungen zu definieren und zu erstellen.

Software für IBM PS/2

Lotus gab bekannt, daß neue Versionen der Programme 1-2-3, Symphony und Freelance für die neuen Computer der PS/2-Familie von IBM auf 3½-Zoll-Laufwerken entwickelt werden. Die neuen Programme sollen die Vorteile des neuen DOS 3.3 nutzen, was sich besonders bei der Datenspeicherung und dem Einsatz der neuen Bildschirme bemerkbar machen wird. Für das neue Betriebssystem /2 wird Lotus das Programm 1-2-3 anpassen.

Cobol für IBM PS/2

IBM hat Micro Focus Cobol für seine neue Generation von Computern – PS/2 mit dem Betriebssystem /2 – und für die bisherigen IBM PCs lizensiert. Das IBM Cobol Version 2.0 soll ab Juni 1987 für DOS 3.2 und DOS 3.3 verfügbar sein. Sobald das neue Betriebssystem allgemein verfügbar ist, werden die Benutzer Cobol Version 2.0 auf Cobol/2 aufrüsten können. Cobol/2 ist ein ANSI '85 Compiler mit eingebautem interaktivem Testsystem.

DIE INTELLIGENTE FESTPLATTEN LÖSUNG...

QUANTUM

...für Ihren PC!
Der I.Q. Kit von Quantum

- 50 und 80 MB formatierte Kapazität
- 5 1/4" halbe Bauhöhe
- incl. SCSI-Hostadapter, Treiber Software und anschlußfertiges Kabel für XT/AT und Kompatible
- 10 MB/sec Transferrate, tacc = 27 ms
- MTBF 25000 Std.
- erweiterungsfähig und SCSI-Schnittstelle
- erschwinglich

Einführungs-Preise

I.Q. 250 Kit	DM 2394,–/St. inkl. MwSt.
I.Q. 280 Kit	DM 2964,–/St. inkl. MwSt.

*Eine Bestellung, die sich lohnt!!!

IST-ELEKTRONIK VERTRIEBS GmbH
Systeme und Peripherie

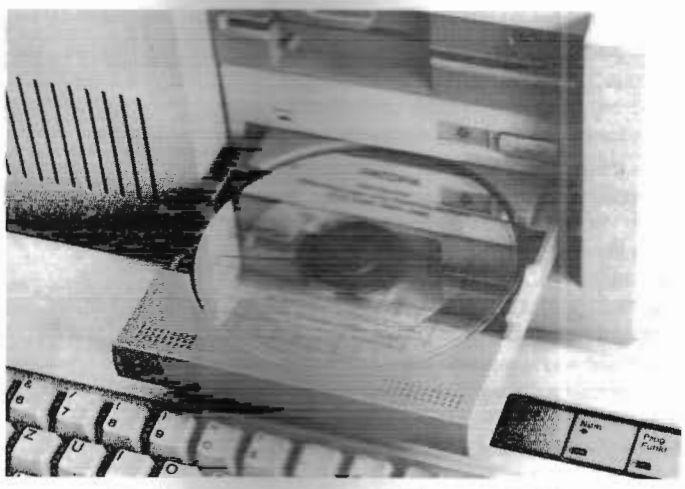
tm 4914

Haupsitz: 8025 Unterhaching
Kapellenstr. 9 · Postfach 11 40
Tel.: 089/6 11 61 51 · Telex: 5 216 650
Fax: 089/6 11 61 20

Büro Nord:
2900 Oldenburg · Bahnhofsplatz 1
Tel.: 04 41/2 59 31 · Telex: 2 5716

Patentdatenbank auf CD-ROM

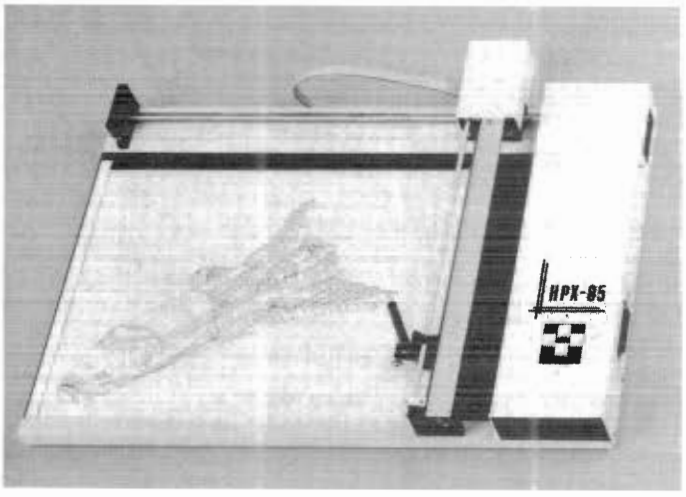
Die Kölner Firma Scientific Consulting, die sich als Dienstleistungsunternehmen für den Einsatz von CD-ROMs betätigt, hat jetzt die Tauglichkeit dieses optischen Speichermediums für Informationen nachgewiesen, die aus Texten und Zeichnungen bestehen. Auf der INFOBASE '87 in Frankfurt zeigte sie eine CD-ROM mit insgesamt 20 000 Offenlegungs- und Patentschriften aus Bereichen wie Physik, Meßtechnik, Optik und Fotografie.
Jedes der gespeicherten Patentdokumente besteht aus bibliografischen Informationen, einer Kurzdarstellung, den Patentzeichnungen in Vektordarstellung sowie dem Volltext der Patentschrift im Umfang von mehreren DIN-A4-Seiten. Außerdem wurden für Demonstrationszwecke komplette Patentdokumente im Faksimileformat auf der CD-ROM erfaßt.
In den Faksimiledokumenten kann wie in einem Buch geblättert werden. Die einzelnen Seiten können sogar über einen Laserdrucker im Originallayout ausgegeben werden. Außerdem ist die direkte Übertragung zu einem Faksimilegerät möglich.
Um derartige CD-ROMs lesen zu können, ist ein IBM-PC-kompatibler Computer mit CD-ROM-Laufwerk (ab 1500 DM erhältlich) erforderlich. Zur Ausgabe der Faksimiledokumente braucht man einen hochauflösenden Grafikbildschirm, wie sie ebenfalls schon für relativ wenig Geld angeboten werden.



Texte und Zeichnungen von Patentschriften auf CD-ROM

Plotten und Digitalisieren für wenig Geld

Um zwei Modelle hat die Firma Habersetzer ihre HPX-Plotter-Serie erweitert. Der HPX-85 ist der Nachfolger des HPX-84. Er erreicht eine Geschwindigkeit von 100 mm/s bei einer Auflösung von 0,025 mm. Ein Puffer von 48 KByte verkürzt die Wartezeiten beim Zeichnen. Das Gerät kostet knapp 1800 DM.
Eine intelligente Lösung des Digitalisierproblems hat man sich beim HPX-86 ausgedacht. Dieses Gerät verhält sich nach dem Einschalten zunächst wie ein ganz normaler Plotter. Auf Knopfdruck läßt es sich jedoch in den Digitalisiermodus schalten. Der Stifthalter wird dann durch ein Fadenkreuz ersetzt, und an den Plotter schließt man die mitgelieferte Maus an. Verschiebt man die Maus jetzt auf dem Tisch, so bewegt sich das Fadenkreuz analog dazu. Dadurch ist es möglich, Konturen nachzufahren und Koordinaten aufzunehmen. Der Plotter, oder besser gesagt der Digitalisierer, überträgt die Daten über eine V24-Schnittstelle zum Computer. Das Datenformat kann man so einstellen, daß es entweder dem einer Microsoft-Maus oder eines Summa-Sketch-Grafiktabletts entspricht. Ausgeben muß man für dieses interessante Gerät rund 2500 DM.



Preiswerte Plotter sind nach wie vor die Spezialität der Firma Habersetzer

Konferenz über Telekommunikation

Die internationale Unternehmensberatung für Informationsverarbeitung und Telekommunikation Pactel bereitet eine internationale Konferenz über Telekommunikation vor. Auf der Veranstaltung, die am 11. und 12. November 1987 in Amsterdam stattfindet, werden vier Themenblöcke behandelt: internationale Situation, Standpunkt der Lieferanten, Überblick über Dienstleistungsangebote, Perspektiven der Anwender. Nähere Informationen gibt es unter 0 69/71 09 26 6 bei Thea Boudreaux.

IBM-Wissenschaftler geehrt

Die Mandelbrot-Menge ist heute jedem Informatiker ein Begriff. Ihr Entdecker, Benoit B. Mandelbrot vom IBM-Forschungszentrum Yorktown Heights wurde jetzt zum Mitglied der National Academy of Sciences in Washington gewählt. Dieselbe Ehre wurde Gerd K. Binnig vom IBM-Labor in Zürich zuteil. Die beiden Wissenschaftler wurden damit für ihre herausragenden Leistungen in der Grundlagenforschung ausgezeichnet. Dr. Mandelbrot ist Entdecker der fraktalen Geometrie, Dr. Binnig erhielt 1986 den Physik-Nobelpreis für seine Arbeit am Rastertunnelmikroskop. Die National Academy of Sciences ist eine Organisation von Wissenschaftlern zur Forschungsförderung. Sie ist auch offizieller Berater der US-Regierung für Wissenschaft und Technik.

Werkzeug Personal Computer

H.P. Blomeyer-Bartenstein, Personal Computer – das intelligente Werkzeug für jedermann, Markt & Technik Verlag AG, Haar, 1985, 53 DM.
ISBN 3-89090-115-8.

Bereits in der zweiten Auflage erscheint das Buch von H.P. Blomeyer-Bartenstein, das dem Leser alles Wichtige und Nützliche, was im Gebrauch von PCs einschließlich des weiten Umfeldes eine Rolle spielt, vermittelt. Umgangssprachlich für den Neuling geschrieben, Sachkenntnisse für den Fachmann aufzeigend, hilft dieses Werk bei der grundsätzlichen Orientierung über Funktion, Einsatzmöglichkeiten und Grenzen von PCs, deren Auswahl sowie die spezielle Verwendung in bestimmten Berufszweigen. Beginnend mit der Einordnung von Personal Computern über seinen prinzipiellen Aufbau legt der Autor die vielfältigen Einsatzgebiete dar. Anschaulich, mit zahlreichen Bildern und Tabellen, wird danach der Aufbau eines PC detailiert dargestellt. Dieses dominierende Kapitel stellt auch für den Fortgeschrittenen eine übersichtliche Fundgrube dar. Der Anschluß verschiedener Geräte an den PC fehlt ebensowenig wie die Erläuterung von Mehrbenutzersystemen einschließlich der Nutzung von Personal Computern zur Datenfernübertragung. Grundlegende Details von Software werden erörtert, schließlich Hinweise auf einzusetzende Programmiersprachen gegeben. Einige Hinweise zum Bezug von Anwendungssoftware, zur möglichst langen Nutzung des PC und eine Marktbersicht runden das Buch ab. Insgesamt ein Buch, das umfassend aufbereitet wurde, dazu verständlich und mit einer Fülle von Literaturangaben versehen ist. *Dr. Th. Weller*

Programmierung des 80286

Von C.Vieillefond, übersetzt aus dem Französoschen von Iris Cantarellas. 509 Seiten, gebunden. Sybex-Verlag, Düsseldorf, 1987, 68 DM.
ISBN 3-88745-668-8

Es sei nur am Rande erwähnt, daß in diesem ins Deutsche übersetzte Buch „Quarz" konsequent amerikanisch als „Quartz" geschrieben wird. Außerdem, daß in einer Abbildung, es ist Abb. 5.6, völlig neuartige geschweifte Logik-Baustein-Symbole benutzt werden, die wohl ein ahnungsloser Zeichner erfunden hat. Wo ist das Positive an diesem Buch? Wer weiß, daß man als normaler Bürger Datenbücher nur unter schwierigen Umständen erhält, der muß dem Sybex-Verlag dankbar sein, daß er die Intel-Unterlagen für die Öffentlichkeit so aufbereitet, daß ein vollständiges Prozessorhanbuch daraus wird. Der Band enthält im ersten Teil die Übersicht über die Hardware-Konstruktion des Prozessors. Dann werden die Speicherverwaltung und die Mechanismen für die virtuelle Adressierung geschildert. Danach geht es in die eigentlichen Eigenschaften des Prozessors: in die Task-Verwaltung und die Mechanismen zur Privileg-Steuerung.
Das Buch enthält danach die Beschreibung der einzelnen Prozessorbefehle wie sie im Handbuch stehen und danach dann noch 17 Programmmbeispiele. Das Buch ist zum Einsteigen gut brauchbar, weil fast alles darin geschildert wird, was für die Einarbeitung notwendig ist. Die Programme sind mit dem Intel-ASM-286 entwickelt. *Ro*

NDR-Klein-Computer zum Nachlesen

Rolf-Dieter Klein: Rechner modular. 423 Seiten, kart. 68 DM. Franzis-Verlag, München, 1987.
ISBN 3-7723-8721-7

Auch für die Computer-Praxis gilt: Was man mit eigenen Augen gesehen hat, bleibt im Gedächtnis haften. Wer aber hinter die Kulissen, sprich das Gehäuse seines Computers blicken will, um die Funktionsweise zu ergründen, wird oft enttäuscht. Da viele Hersteller spezielle integrierte Bausteine verwenden, ist man oft hinterher nicht klüger als zuvor.
Den umgekehrten – und weitaus effektiveren – Weg geht Rolf-Dieter Klein beim Bau seines „NDR-Klein-Computers". In seinem Begleitbuch zur gleichnamigen Sendereihe im Bayerischen Fernsehen gibt der Autor praktische Anleitungen zum Selbstbau eines Z80-Computersystems. Das Buch zeichnet sich vor allem durch seinen didaktischen Aufbau aus. Fragen am Ende eines jeden Kapitels geben dem Leser die Chance, zu prüfen, ob die Informationen der vorhergehenen Seiten auch wirklich „gespeichert" wurden.
Ausgerüstet mit grundlegenden Erklärungen zur Spannungsversorgung und zur Digitaltechnik bereitet es nun keine Schwierigkeiten mehr, quasi als Einstieg, einen Single Board Computer aufzubauen. Stößt man schließlich an dessen Speichergrenzen (16 KByte), erlaubt das modulare Konzept die Erweiterung durch eine Z80-CPU- bzw. CPU 64180-Karte. Zum Betrieb dieser Karten benötigt man eine 64-KByte-Speicher-Karte, die vorbildlich erklärt wird. Zum professionellen CP/M-Gerät wird der NDR-Klein-Computer durch einen Floppy-Anschluß.
Monika Kagerbauer.

Das große MS-DOS-Profi-Arbeitsbuch

Von Dieter Smode. 404 Seiten mit 67 Abbildungen, Tabellen und Listings. Lwstr.-kart. 68 DM. Franzis-Verlag, München.
ISBN 3-7723-8681-4

Die zum Thema MS-DOS/PC-DOS erhältliche Literatur beschäftigt sich in der Regel immer wieder mit dem gleichen Thema: der Schnittstelle zwischen Mensch und Maschine. Beim MS-DOS-Profi-Arbeitsbuch war es jedoch das Ziel des Autors, alle Belange von MS-DOS und PC-DOS zu berücksichtigen und eine umfassende Beschreibung der Schnittstelle vom Anwenderprogramm zum Betriebssystem zu geben. Dies bedeutet, daß die zahlreichen Betriebssystemaufrufe detailliert dargestellt werden, um dem Programmierer und auch dem fortgeschrittenen Anwender das Leben zu erleichtern. Ausgehend von der Version 2.1x werden alle Änderungen der Folgeversionen bis einschließlich DOS 3.20 beschrieben. Es ist erfreulich, daß der Autor auch nicht dokumentierte, aber von Microsoft sehr wohl verwendete, Funktionsaufrufe erläutert. Interessant ist die Beschreibung des Funktionsaufrufs 34H (Get IN_DOS_FLAG). Mit diesem Aufruf kann die Adresse des Flags ermittelt werden, das anzeigt, ob zu diesem Zeitpunkt DOS-Aufrufe erlaubt sind.
Auf über 70 Seiten schildert der Autor die Erstellung von Programmen und die Programmierwerkzeuge unter DOS. Insbesondere geht er auf die DOS-Schnittstellen zu den Hochsprachen ein. Abgerundet wird dieses Buch durch ausführliche Erörterungen zu Fragen der Kompatibilität und Portabilität von DOS-Programmen. *St*

Dr. J. Reinkenhof

3D-Darstellung von Funktionen

Das hier vorgestellte Programm dient zur dreidimensionalen Darstellung einer Funktion Z(X,Y) unter Einhaltung der Sichtbarkeitsbedingung. Dieses rechenintensive Problem läßt sich mit dem schnellen GFA-Basic-Interpreter auf dem Atari ST in einer für durchschnittliche Anwendungen akzeptablen Rechenzeit lösen.

Die grafische Darstellung von Funktionen mit zwei unabhängigen Veränderlichen geschieht meist mit Kurvenscharen, wobei eine Veränderliche unterschiedliche Parameterwerte durchläuft. Ein solches Diagramm bietet zwar Vorteile beim Ablesen von Funktionswerten, gibt dem Betrachter jedoch – im Falle einer komplizierten Funktion – keinen anschaulichen Einblick in deren qualitatives Verhalten. Bei der Lektüre technischer und wissenschaftlicher Zeitschriften stößt man daher in zunehmendem Maße auf dreidimensionale, computergrafische Darstellungen, welche durch axonometrische Projektion gewonnen wurden.

Derartige „Funktionslandschaften" kommen der räumlichen Vorstellung sehr entgegen und eignen sich gut zur Darstellung topologischer Merkmale der Funktionsoberfläche. Noch vor einigen Jahren waren solche grafischen Darstellungen – unter Einschluß der Sichtbarkeitsbedingung – in zufriedenstellender Qualität nur durch den Einsatz eines Großrechners zu erhalten. Der Grund dafür lag nicht in einem aufwendigen Algorithmus, sondern in der Notwendigkeit, bei der Berechnung der Sichtbarkeit jedes einzelnen Bildpunktes mit sehr kleinen Schrittweiten (hoher Diskretisierung) arbeiten zu müssen. Bei zu grober Diskretisierung heben sich zum Beispiel die Randkonturen eines im Vordergrund sichtbaren „Funktionsberges" vom Hintergrund nicht in einer dem Auge vertrauten Weise ab. Es liegt also ein rechenintensives Problem vor, bei dessen Lösung ein 16-/32-Bit-Mikrocomputer der neuen Generation seine Leistungsfähigkeit beweisen kann.

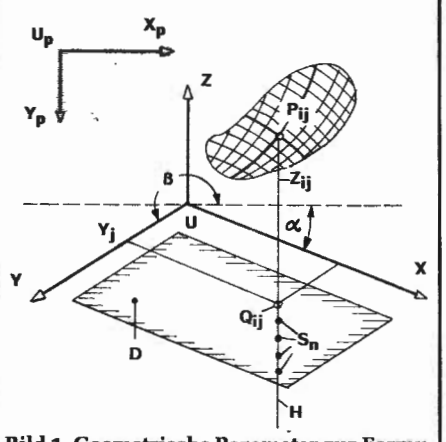


Bild 1. Geometrische Parameter zur Formulierung der Sichtbarkeit einer Funktionsfläche in axonometrischer Darstellung

Die Sichtbarkeit von Bildpunkten

Zum Verständnis des hier vorgestellten Programms wird eine kurze Erläuterung des Algorithmus zur Bestimmung der Sichtbarkeit beliebiger Bildpunkte gegeben. *Bild 1* dient zur Veranschaulichung der geometrischen Zusammenhänge. Bei der orthogonalen Axonometrie ist die Projektionsrichtung senkrecht zur Zeichenebene orientiert. Zur Konstruktion des Funktionsbildes wird ein beliebig mit den Winkeln α und β festlegbares, rechtwinkeliges Koordinatensystem X,Y,Z eingeführt, dessen Ursprung U nicht in der Zeichenebene liegt. Bezüglich der Lage der Achsen gibt es bevorzugte Sonderfälle (als Kavalierperspektive bezeichnet man z. B. die Anordnung $\alpha = 0$ Grad, $\beta = 45$ Grad).

Wir betrachten in Bild 1 eine Funktion Z(X,Y), welche über einem Gebiet D (schraffierte Fläche) dargestellt werden soll. Nach Diskretisierung der Funktion in den Koordinaten X und Y läßt sich die Funktionsoberfläche durch Polygonzüge darstellen. Die Lage des Gebietes D innerhalb der X,Y-Ebene sei beliebig. Geprüft werden soll nun die Sichtbarkeit eines Gitterpunktes P_{ij}, welcher durch das Wertetripel X_i,Y_j,Z_{ij} bestimmt wird.

Der Punkt P_{ij} ist nur dann sichtbar, wenn ein von ihm ausgehender, senkrecht auf die Bildebene (Zeichenebene) zulaufender Sichtstrahl nicht in das „Funktionsgebirge" eindringt, also ungestört beim Beobachter in der Bildebene ankommt. Man denkt sich nun eine zur Bildebene senkrecht stehende und zur Z-Achse parallel verlaufende Hilfsebene H, welche den Gitterpunkt P_{ij} und den auf dem Gebiet D liegenden Fußpunkt Q_{ij} enthält. Der Schnitt dieser Hilfsebene H mit der X,Y-Ebene erzeugt auf dem Gebiet D einen Spurabschnitt, dessen vorderen Teil man mit Spurpunkten $\{S_n\}$ belegt, deren Koordinaten X_n,Y_n seien. Aus der Anschauung folgt zunächst, daß nur die über den Spurpunkten $\{S_n\}$ liegenden Funktionswerte $\{Z_n\}$ die Sichtbarkeit von P_{ij} entscheiden.

Eine endgültige Aussage darüber ist jedoch nur durch Umrechnung in das Koordinatensystem der Bildebene mit den Koordinatenachsen X_p,Y_p möglich, wobei der Koordinatenursprung U_p (entsprechend den grafischen Sprachelementen des GFA-Basic) in der linken oberen Bildschirmecke liegt. Der Index p soll daran erinnern, daß die Maßeinheit in diesem System 1 Pixel beträgt. Diese Schirmkoordinaten X_p,Y_p lassen sich mit Hilfe linearer Transformationsgleichungen aus dem Koordinatentripel X_i,Y_j,Z_{ij} berechnen. Einzelheiten dazu entnehme man dem Programmlisting (*Bild 2*). Der Gitterpunkt P_{ij} erzeugt auf der Bildebene (Schirm) nur dann ein sichtbares Bild, wenn für alle n innerhalb des Gebietes D die Ungleichung

$$Y_p(X_n,Y_n,Z_n) > Y_p(X_i,Y_j,Z_{ij})$$

gilt. Anderenfalls ist P_{ij} unsichtbar hinter einem „Funktionsgebirge" versteckt. Nur bei einer hinreichend groß gewählten Anzahl n von Spurpunkten $\{S_n\}$ gelingt, im Falle teilweise verdeckter Funktionsoberflächen, eine für das Auge befriedigende axonometrische Darstel-

```
' ------------------------------------------
' -   Programm DREID; GFA-BASIC (Atari ST)  -
' -   3D-Darstellung einer Funktion Z=Z(X,Y) -
' -   Programmautor: Dr.J. Reinkenhof,Heilbronn -
' ------------------------------------------
Funktion:
Deffn Z(X,Y)=X*X+Y*Y-1                          !  beliebiges Z(X,Y)
' ------------------------------------------------------------------
Deffn Demo(X,Y)=Sin((X*X+Y*Y)^0.5)/(X*X+Y*Y)^0.5+0.2 ! nicht aendern -
' ------------------------------------------------------------------
Gosub Eingabedaten
Gosub Minmax
Gosub Konstanten
Gosub Achsenkreuz
Gosub Rahmen
Gosub Kurven
End
Rem  ---------------  Funktionen und Unterprogramme  ---------------
Deffn Xp(X,Y)=Xurp+K1*X+K2*Y        ! Transformation in pixel-Koord.
Deffn Yp(X,Y,Z)=Yurp+L1*X+L2*Y+L3*Z  !              "
Procedure Eingabedaten
  ' ------------  Data-Werte nicht aendern (Demonstration) ----------
  Data 30,200,30,200,30,200,2,1,1,0
  Data -20,55,1.1,1.1,1.1,-10,10,-10,10,1,2,0.45
  Read Xminp,Xmaxp,Yminp,Ymaxp,Zminp,Zmaxp,GX,P,Gitter,Markierung ! --
  Read A,B,Mx,My,Mz,Xmin,Xmax,Ymin,Ymax,P,Ww,Ey                  ! --
  ' ------------------------------------------------------------------
  Clearw 2
  Fullw 2
  Input "Demonstrationsbeispiel-->(0)   neue Funktion Z(X,Y)-->(1) ",De
  If De=0
    Goto Marke1
  Endif
  Input "z(x,y) definiert:  ja-->(1) nein-->(0)",B
  If B=0 Then
    Input "Funktion Z(X,Y) bei Sprungmarke 'Funktion' eingeben!",Bb
  Endif
  ' x-Achse und y-Achse moeglichst in benachbarten Quadranten
  Input "Winkel x-Achse/Horizontale (innerhalb +-90 Grad;z.B.:30 ) ?",A
  Input "Winkel y-achse/Horizontale (z.B.:130 Grad) ?",B
  Input "Verkuerzung x-Achse (Norm: 1) ?",Mx
  Input "Verkuerzung y-Achse (Norm: 1) ?",My
  Input "Verkuerzung z-Achse (Norm: 1) ?",Mz
  Input "Xmin?",Xmin
  Input "Xmax?",Xmax
  Input "Ymin?",Ymin
  Input "Ymax?",Ymax
  Input "Ursprung von Z(X,Y) auf Schirm (0=unten;1=oben; z.B.:0.4)?",Ey
  Input "Sichtbedingung einhalten: ja-->(1) nein-->(0) ?",P
  If P=1 Then
    Input "Diskretisierung:grob-->(1) genau-->(2) sehr genau-->(3)?",GX
  Endif
  Input "Linien parallel zu: X-->(1)  Y-->(2)   X und Y-->(3) ?",Gitter
  Input "Achsenskalierung:  ja--(1)   nein-->(0)  ?",Markierung
  Marke1:
  Print "Bitte auf 3D-Funktion warten !"
Return
' ------------------------------------------------------------------
Procedure Konstanten
  A=A*Pi/180
  B=B*Pi/180
  If Sgn(A)<0 Then
    A=2*Pi+A  ! Umrechnung ins Positive
  Endif
  If Sgn(B)<0 Then
    B=2*Pi+B  !          "
  Endif
  Xurp=180  ! Koordinatenursprung (pixel)
  Yurp=400*(1-Ey)  !    "
  Nx=150   ! Norm: Anzahl Pixel fuer(Xmax-Xmin)
  Ny=150   ! Norm         "        (Ymax-ymin)
  Nz=150   ! Norm         "        (Zmax-Zmin)
  Rem   K1,K2,L1,L2,L3 --> Koeffizienten der Transformationsgln.
  K1=Nx*Mx*Cos(A)/Norm_x
  K2=Ny*My*Cos(B)/Norm_y
  L1=-Nx*Mx*Sin(A)/Norm_x
  L2=-Ny*My*Sin(B)/Norm_y
  L3=-Nz*Mz/Norm_z
  If GX=1 Then
    SxX=25    ! Diskretisierungsparameter (aenderbar)
    SyX=25    !              "                "
  Endif
  If GX=2 Then
    SxX=50    !              "                "
    SyX=50    !              "                "
  Endif
  If GX=3 Then
    SxX=75    !              "                "
    SyX=75    !              "                "
  Endif
  Dx=(Xmax-Xmin)/(SxX-1) ! Schrittweite x-Achse
  Dy=(Ymax-Ymin)/(SyX-1) ! Schrittweite y-Achse
  Dxs=Dx                 ! Schrittweite fuer Spurpunkte
  Dys=Dxs*K1/Abs(K2)     ! Schrittweite fuer Spurpunkte
  NmaxX=Int((Xmax-Xmin)/Dxs)+1 ! Max. Schrittzahl auf Spur
  If A>Pi Then
    V_x=1                ! Schrittsteuerparameter (spur)
  Else
    V_x=-1               !        "
  Endif
  If B>Pi Then
    V_y=1                !        "
  Else
    V_y=-1               !        "
  Endif
  If A<Pi Then
    If B<Pi Then
      CX=1     !  C --> Kennzahl fuer Achsanordnung
    Else
      CX=2
    Endif
  Endif
  If A>Pi
    If B>Pi Then
      CX=3
    Else
      CX=4
    Endif
  Endif
Return
' ------------------------------------------------------------
Procedure Achsenkreuz
  Openw 2
  Clearw 2
  Xx=1.2*Nx*Cos(A)
  Xy=1.2*Nx*Sin(A)
  Yx=1.2*Ny*Cos(B)
  Yy=1.2*Ny*Sin(B)
  Zy=1.25*Nz
  Defline 1,2,0,1                          !Pfeil Linienende
  Line Xurp,Yurp,Xurp,Yurp-Zy              !Z-Achse
  Line Xurp,Yurp,Xurp+Xx,Yurp-Xy           !X-Achse
  Line Xurp,Yurp,Xurp+Yx,Yurp-Yy           !Y-Achse
  Text Xurp,Yurp-1.05*Zy,"Z"
  Text Xurp+1.1*Xx,Yurp-1.1*Xy,"X"
  Text Xurp+1.1*Yx,Yurp-1.1*Yy,"Y"
  ' Angabe der Variablenintervalle:
  Print At(50,15);"X-->[";Using "+#.#^^^^",Xmin
  Print At(63,15);","
  Print At(64,15);Using "+#.#^^^^",Xmax
  Print At(72,15);"]"
  Print At(50,16);"Y-->[";Using "+#.#^^^^",Ymin
  Print At(63,16);","
  Print At(64,16);Using "+#.#^^^^",Ymax
  Print At(72,16);"]"
  Print At(50,17);"Z-->[";Using "+#.#^^^^",Zmin
  Print At(63,17);","
  Print At(64,17);Using "+#.#^^^^",Zmax
  Print At(72,17);"]"
  Print At(66,20);"mc-soft"
```

Bild 2. GFA-Basic-Programm zur dreidimensionalen Darstellung von Funktionen

```
  ' Achsenskalierung:
  Defline 1,2,0,0
  Hx=Nx*Xmax/Norm_x*Mx/10  ! Hx,Hy,Hz -->Skalierungsab-
  Hy=Ny*Ymax/Norm_y*My/10  ! schnitte auf X-,Y-,Z-Achse
  Hz=Nz*Zmax/Norm_z*Mz/10
  If Markierung=1 Then
    For U=1 To 10
      Xx_p=Xurp+U*Hx*Cos(A)
      Xy_p=Yurp-U*Hx*Sin(A)
      Yx_p=Xurp+U*Hy*Cos(B)
      Yy_p=Yurp-U*Hy*Sin(B)
      Zx_p=Xurp
      Zy_p=Yurp-U*Hz
      Line Xx_p,Xy_p,Xx_p+Nx/50*Cos(B),Xy_p+Nx/50*Sin(B)
      Line Yx_p,Yy_p,Yx_p-Ny/50-Cos(A),Yy_p+Ny/50*Sin(A)
      Line Zx_p,Zy_p,Zx_p+Nx/50,Zy_p
    Next U
  Endif
Return
Procedure Rahmen
  '     Zeichnen der Eckwerte u. der Berandung
  '     des Gebietes D in der x,y-Ebene
  '     P1...P4 -->Eckpunkte Gebiet D
  '     P5...P8 -->Funktionspunkte ueber P1...P4
  X1=Fn Xp(Xmin,Ymax)
  Y1=Fn Yp(Xmin,Ymax,0)
  X2=Fn Xp(Xmax,Ymax)
  Y2=Fn Yp(Xmax,Ymax,0)
  X3=Fn Xp(Xmax,Ymin)
  Y3=Fn Yp(Xmax,Ymin,0)
  X4=Fn Xp(Xmin,Ymin)
  Y4=Fn Yp(Xmin,Ymin,0)
  X5=X1
  If De=1 Then
    Z1_5=Fn Z(Xmin,Xmax)
  Endif
  If De=0 Then
    Z1_5=Fn Demo(Xmin,Xmax)
  Endif
  Y5=Fn Yp(Xmin,Ymax,Z1_5)
  X6=X2
  If De=1 Then
    Z2_6=Fn Z(Xmax,Ymax)
  Endif
  If De=0 Then
    Z2_6=Fn Demo(Xmax,Ymax)
  Endif
  Y6=Fn Yp(Xmax,Ymax,Z2_6)
  X7=X3
  If De=1 Then
    Z3_7=Fn Z(Xmax,Ymin)
  Endif
  If De=0 Then
    Z3_7=Fn Demo(Xmax,Ymin)
  Endif
  Y7=Fn Yp(Xmax,Ymin,Z3_7)
  X8=X4
  If De=1 Then
    Z4_8=Fn Z(Xmin,Ymin)
  Endif
  If De=0 Then
    Z4_8=Fn Demo(Xmin,Ymin)
  Endif
  Y8=Fn Yp(Xmin,Ymin,Z4_8)
  If B>Pi Then
    Line X1,Y1,X5,Y5   !  Eckwert
    Line X1,Y1,X2,Y2   !  Randlinie Gebiet D
    Line X2,Y2,X6,Y6   !  Eckwert
  Endif
  If A>Pi Then
    Line X2,Y2,X6,Y6
    Line X2,Y2,X3,Y3
    Line X3,Y3,X7,Y7
  Endif
  If B<Pi Then
    Line X3,Y3,X7,Y7
    Line X3,Y3,X4,Y4
    Line X4,Y4,X8,Y8
  Endif
  If A<Pi Then
    Line X4,Y4,X8,Y8
    Line X4,Y4,X1,Y1
    Line X1,Y1,X5,Y5
  Endif
Return
' ------------------------------------------------------
Procedure Kurven
  ' Zeichnen der "Funktionslandschaft"
  Delta_x=1
  Delta_y=1
  If Gitter=2 Then
    Delta_y=Sy%-1
  Endif
  For J%=0 To Sy%-1 Step Delta_y
    For I%=0 To Sx%-1 Step Delta_x
      Berg%=0
      Xi=Xmin+I%*Dx
      Yj=Ymin+J%*Dy
      If De=1 Then
        Zij=Fn Z(Xi,Yj)
      Endif
      If De=0 Then
        Zij=Fn Demo(Xi,Yj)
      Endif
      If Xi=Xmin Then
        Xpalt=Fn Xp(Xi,Yj)
        Ypalt=Fn Yp(Xi,Yj,Zij)
      Endif
      If P=1 Then
        Gosub Sicht
      Endif
      If Berg%=0 Then
        Rem  Schnittlinien von Ebenen !! zur x,z-Ebene:
        Line Xpalt,Ypalt,Fn Xp(Xi,Yj),Fn Yp(Xi,Yj,Zij)
      Endif
      Xpalt=Fn Xp(Xi,Yj)
      Ypalt=Fn Yp(Xi,Yj,Zij)
    Next I%
  Next J%
  If Gitter=1 Then
    Delta_x=Sx%-1
  Endif
  If Gitter=2 Then
    Delta_y=1
  Endif
  For I%=0 To Sx%-1 Step Delta_x
    For J%=0 To Sy%-1 Step Delta_y
      Berg%=0
      Xi=Xmin+I%*Dx
      Yj=Ymin+J%*Dy
      If De=1 Then
        Zij=Fn Z(Xi,Yj)
      Endif
      If De=0 Then
        Zij=Fn Demo(Xi,Yj)
      Endif
      If Yj=Ymin Then
        Xpalt=Fn Xp(Xi,Yj)
        Ypalt=Fn Yp(Xi,Yj,Zij)
      Endif
      If P=1 Then
        Gosub Sicht
      Endif
      If Berg%=0 Then
        '  Schnittlinien von Ebenen !! zur y,z-Ebene:
        Line Xpalt,Ypalt,Fn Xp(Xi,Yj),Fn Yp(Xi,Yj,Zij)
      Endif
      Xpalt=Fn Xp(Xi,Yj)
      Ypalt=Fn Yp(Xi,Yj,Zij)
    Next J%
  Next I%
Return
' ------------------------------------------------------
Procedure Minmax
  '   Berechnung der Extremwerte von Z(X,Y)
  Zmin=1.0E+90 ! "unendlich"
  Zmax=0
  Rx%=30 !bei kompliziertem F(x,y):  50 setzen
  Ry%=30 !            "          :  50 setzen
  Dx=(Xmax-Xmin)/(Rx%-1)
  Dy=(Ymax-Ymin)/(Ry%-1)
  For Xi=Xmin To Xmax Step Dx
    For Yj=Ymin To Ymax Step Dy
      If De=1 Then
        Zij=Fn Z(Xi,Yj)
      Endif
      If De=0 Then
        Zij=Fn Demo(Xi,Yj)
      Endif
      Zmin=Min(Zmin,Zij) ! Minimum
      Zmax=Max(Zmax,Zij) ! Maximum
    Next Yj
  Next Xi
  '  Berechnung der Normierungsgroeen;
  If Xmin>=0 Then
    Norm_x=Xmax
  Else
    Norm_x=Xmax-Xmin
  Endif
  If Ymin>=0 Then
    Norm_y=Ymax
  Else
    Norm_y=Ymax-Ymin
  Endif
  If Zmin>=0 Then
    Norm_z=Zmax
  Else
    Norm_z=Zmax-Zmin
  Endif
  If Zmin=Zmax Then
    Norm_z=Zmax
  Endif
Return
' ------------------------------------------------------
Procedure Sicht
  '  Geprueft wird, ob der Punkt Pij=P(Xi,Yj)
  '  s i c h t b a r  oder  u n s i c h t b a r  ist
  For N%=1 To Nmax%
    If N%>1 Then
      Goto Spur
    Endif
    Exit If C%=1 And I%=0
    Exit If C%=1 And J%=0
    Exit If C%=2 And I%=0
    Exit If C%=2 And J%=Sy%-1
    Exit If C%=3 And I%=Sx%-1
    Exit If C%=3 And J%=Sy%-1
    Exit If C%=4 And I%=Sx%-1
    Exit If C%=4 And J%=0
    Spur:
    Xn=Xi+V_x*Dxs*N%    ! Koordinate Spurpunkt
    Yn=Yj+V_y*Dys*N%    !          "
    If De=1 Then
      Zn=Fn Z(Xn,Yn)
    Endif
    If De=0 Then
      Zn=Fn Demo(Xn,Yn)
    Endif
    If Fn Yp(Xn,Yn,Zn)<Fn Yp(Xi,Yj,Zij) Then
      Berg%=1       ! Punkt Pij unsichtbar
    Endif
    Exit If Berg%=1 ! keine Sicht
    Exit If Xn>=Xmax
    Exit If Yn>=Ymax
    Exit If Xn<=Xmin
    Exit If Yn<=Ymin
  Next N%
Return
```

lung. Der Rechenprozeß ist also aufwendig, weil für jedes P_{ij} neue Funktionswerte $\{Z_n\}$ berechnet werden müssen, die in $\{Z_{ij}\}$ nicht enthalten sind.

Mit Maßstabsfaktoren M_X, M_Y, M_Z läßt sich das Bild in den drei Achsrichtungen beliebig verändern. Der Normalfall beträgt $M_X = M_Y = M_Z = 1$. Bei kleineren Werten findet eine Stauchung, bei größeren eine Streckung statt.

Programmanwendung und Beispiel

Das Programm wurde in der mittlerweile weit verbreiteten Sprache GFA-Basic [1, 2] geschrieben, einem Hochgeschwindigkeitsinterpreter für den Atari ST.
Sollte die Rechenzeit als zu langsam empfunden werden (die Beobachtung des Bildaufbaus kann allerdings durchaus anregend sein), so kann die Programmausführung mit dem nun ebenfalls verfügbaren GFA-Basic-Compiler [3] beschleunigt werden. Der strukturierte Aufbau des Programms erleichtert gegebenenfalls das Umschreiben in eine andere Sprache. Auf eine ausführliche Beschreibung des Programms sei hier verzichtet, weil die im Hauptprogramm aufgerufenen sechs Unterprogramme vom Umfang her überschaubar sind und mit Kommentaren versehen wurden. Einige Hinweise zur Handhabung des Programms seien jedoch gegeben.

Die darzustellende Funktion Z(X,Y) muß als Deffn Z(X,Y) im Hauptprogramm abgelegt werden. Eine spezielle Programmoption erlaubt die Darstellung einer zweiten, ebenfalls von X und Y abhängigen, beliebig austauschbaren Demonstrationsfunktion, welche unter Deffn Demo(X,X) definiert wird. Es gibt keine Restriktionen bezüglich des Vorzeichens der darzustellenden Funktionen und ihrer Darstellungsbereiche. Auch treten keine Skalierungsprobleme auf. Gefordert wird lediglich, daß es sich um eine einwertige Funktion handelt.

Die Darstellung einer mehrwertigen Funktion (z. B. einer Kugeloberfläche) ist möglich, erfordert jedoch einige Programmänderungen. Der Winkel α und β sollten so gewählt werden, daß die Projektion der Achsen X und Y in der Bildebene in benachbarten Quadranten liegen. Das Programm erlaubt auf Wunsch auch die Skalierung der Achsen (Markierungsabschnitte sind 1/10 des jeweiligen Maximalwertes). Bezüglich der Diskretisierung und damit der Einhaltung der Sichtbarkeitsbedingung kann man folgende drei Optionen wählen: grob, genau, sehr genau. Eine weitere Option läßt auch das Zeichnen von Bildern unter Nichteinhaltung der Sichtbarkeitsbedingung zu. Die letztgenannte Darstellung wird (vermutlich in Ermangelung geeigneter Software) häufig angewendet.

Das Programm erlaubt ferner, wahlweise ausschließlich Schnittlinien in X- oder in Y-Richtung zu zeichnen. Häufig löst eine solche Darstellung beim Betrachter eine bessere räumliche Vorstellung aus im Vergleich zur Projektion eines (im Gebiet D orthogonalen) Gitters auf die Funktionsoberfläche. Durch die geeignete Wahl aller dieser Parameter hat es der Anwender jedenfalls weitgehend in der Hand, ob er sich nur einen schnellen Einblick in den Funktionsverlauf verschaffen will, oder ein veröffentlichungsreifes Bild erzeugen möchte.

Bild 3 zeigt das mit dem Programm errechnete Bild einer gedämpften Schwingung, welche als gutes Testbeispiel für

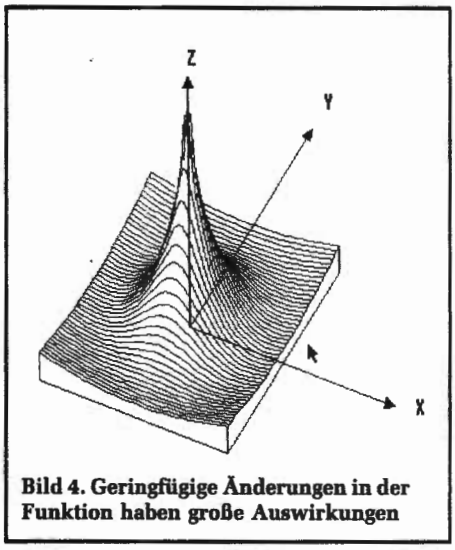


Bild 4. Geringfügige Änderungen in der Funktion haben große Auswirkungen

3D-Darstellungen gilt (eine ähnliche Schwingungserscheinung entsteht z. B. durch punktförmige Störung einer glatten Wasseroberfläche). Es wurde der Diskretisierungsgrad „genau" gewählt. Lokale Darstellungsfehler sind nur am Fuße des Berges erkennbar. Die Rechenzeit beträgt, bei Einsatz des GFA-Basic-Interpreters, immerhin noch 265 Sekunden. Nahezu eine Verdoppelung dieser Zeit ergibt sich beim Zeichnen eines Oberflächengitters. Dem GFA-Basic-Interpreter sind hier offenbar Grenzen gesetzt. Andererseits sind Rechenzeiten in dieser Größenordnung für grafische Darstellungen bei vielen Anwendungen in Naturwissenschaft und Technik durchaus vertretbar.

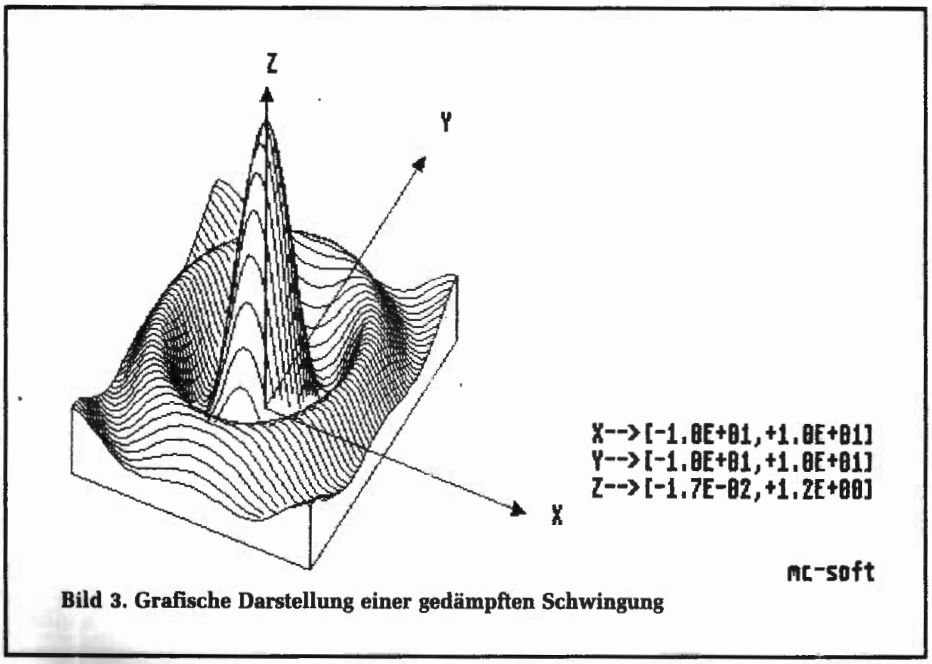


Bild 3. Grafische Darstellung einer gedämpften Schwingung

Literatur

[1] Pernsteiner, P.: GFA-Basic. mc 2/87, S. 84...89.

[2] Führer zu GFA-Basic. Data Becker-Verlag, Düsseldorf, 1986.

[3] Litzkendorf: Das große GFA-Basic-Buch. Data Becker-Verlag, Düsseldorf, 1986.

Manfred Stede

Programmier-Umgebung Dwim und Korr

Ein „Do what I mean"-Programm in muLisp-85

Die Schaffung einer menschenfreundlichen Benutzeroberfläche ist auch eines der Anliegen der künstlichen Intelligenz. Daß man gerade mit KI-Sprachen recht elegante Lösungen hierfür finden kann, zeigt dieser Beitrag.

Angenommen, man hat eine Lisp-Funktion eingegeben, startet die Ausführung und muß nun feststellen, daß man sich an einer Stelle vertippt hat:

```
(DEFUN TEST (...
...
(REVRSE LISTE)
...
```

Der Interpreter beklagt eine „undefined function" bei (REVRSE LISTE). Mit der im Folgenden vorgestellten Funktion gibt man – statt alles neu einzutippen oder zu editieren – nur (KORR TEST-FUNKTION) ein, muLisp verbessert dann den Tippfehler selbsttätig und zeigt die korrigierte Fassung an. Ist man damit zufrieden, kann man die Funktion mit (EVAL ++) gleich wieder starten. Letzteres funktioniert, weil muLisp die letzten drei Benutzereingaben an die Symbole +, ++ und +++ bindet, so daß Eingaben eine Zeitlang erhalten bleiben.

Die nützliche Korrektur-Funktion ist jedoch nicht im Lisp enthalten, sie bildet einen wesentlichen Bestandteil der „do what I mean"-Schnittstelle. Dieser Begriff steht für eine Benutzeroberfläche, die sich nicht gleich an jedem Eingabefehler aufhängt.

```
(defun uebersetze (lambda (wort)
  (pack (doubletten
    (cons (car (unpack wort))
          (remove '*
             (mapcar 'tausche (cdr (unpack wort))) 'equal))))))

(defun tausche (lambda (buchstabe)
  ((member buchstabe '(f p v)) 'b)
  ((member buchstabe '(g j k q s x z)) 'c)
  ((equal buchstabe 't) 'd)
  ((equal buchstabe 'n) 'm)
  ((member buchstabe '(a e i o u y w h)) '*)
  buchstabe))

(defun doubletten (lambda (wortlst)
  ((and (not (null wortlst))
        (not (equal (car wortlst) (cadr wortlst))))
    (cons (car wortlst) (doubletten (cdr wortlst))))
  ((not (null wortlst))
    (cons (cadr wortlst) (doubletten (cddr wortlst))))))


$ (uebersetze 'textverarbeitung)
TCDBRBDMC

$ (uebersetze 'programm)
PRCRM

$ (uebersetze 'weltmeisterschaft)
WLDMCDRCBD
```

Bild 1. Der Kern des Programmes: In der Funktion tausche werden Buchstaben durch Phoneme ersetzt

Phonem-Darstellung

Im vorliegenden Programm wird dazu die schon in mc 6/84 auf Seite 49 vorgestellte phonetische Suche verwendet, mit der ähnlich klingende Worte ineinander umgewandelt werden können.

Um langes Suchen dieses mc-Heftes zu ersparen, seien hier nochmals kurz die Umwandlungs-Regeln aufgeführt:

A, E, I, O, U, Y, W, H	werden entfernt
B, F, P, V	werden zu B
C, G, J, K, Q, S, X, Z	werden zu C
D, T	werden zu D
M, N	werden zu M

Der jeweils erste Buchstabe eines Wortes wird nicht verändert. Die Funktion TAUSCHE (*Bild 1*) enthält die entsprechenden Regeln, wobei jedoch bei A...H provisorisch ein Stern („Asterisk") eingesetzt wird. Mit der letzten Zeile der Funktion werden die Buchstaben L und R ohne Änderung zurückgeliefert.

Die übergeordnete Funktion UEBERSETZE zerlegt das fragliche Wort per UNPACK in einelne Zeichen, anschließend wird TAUSCHE mittels MAPCAR auf jedes einzelne dieser Zeichen angewendet. Die resultierende Liste ist nun das Argument von REMOVE, diese Funktion trennt sämtliche Sternchen heraus. In DOUBLETTEN werden dann alle zweifach hintereinander auftretenden Buchstaben entfernt, anschließend wird alles zusammengepackt (PACK) – und fertig ist die Phonemdarstellung.

DOUBLETTEN ist eine Übung in Listen-Manipulation, die hier nicht weiter vertieft werden soll. Um das Verständnis zu erleichtern, zeigt Bild 1 neben den benötigten Funktionen auch ein paar kleine Beispiele.

```
(defun dwimadd (lambda (kommandoliste)
  ((null kommandoliste) t)
  ((member (uebersetze (car kommandoliste)) dwimlist*)
    (prin1 "Phonem-Stamm von ")
    (prin1 (car kommandoliste))
    (print " bereits vorhanden, Wort wird ignoriert")
    (dwimadd (cdr kommandoliste)))
  (setq dwimlist (cons (car kommandoliste) dwimlist))
  (setq dwimlist* (cons (uebersetze (car kommandoliste))
                        dwimlist*))
  (dwimadd (cdr kommandoliste)) ))

(setq dwimlist nil)
(setq dwimlist* nil)


$ (dwimadd '(length position remove))
T

$ dwimlist
(REMOVE POSITION LENGTH)

$ dwimlist*
(RMB PCDM LMCD)

$ (dwimadd '(huhu findstring haha))
$Phonem-Stamm von $haha$ bereits vorhanden, Wort wird ignoriert$
T

$ dwimlist
(FINDSTRING HUHU REMOVE POSITION LENGTH)
```

Bild 2. Die Phonem-Liste der Schlüsselworte wird mit DWIMADD automatisch erweitert

Die Vorteile

Der Nutzen dieser Methode besteht darin, daß ähnliche Worte auf die selbe Phonem-Darstellung abgebildet werden.

So haben beispielsweise „Computer" und „Compter" das gleiche Abbild, weshalb letzteres als Tippfehler des ersten identifiziert werden kann (und umgekehrt). Um damit praktisch zu arbeiten, legt man alle Worte, für die die Fehlererkennung funktionieren soll, in einer Liste ab. Eine zweite Liste enthält die korrespondierenden Phonem-Darstellungen, die natürlich per Programm erzeugt werden.

Nach Eingabe eines Kommandos wird jedes Wort in die Phonem-Form übersetzt und geprüft, ob es in der Phonem-Liste enthalten ist. Wenn ja, so überschreibt man die Eingabe einfach mit dem ensprechenden Wort aus der Kommando-Liste. Es wird also jedes gespeicherte Wort ersetzt, sei es nun korrekt geschrieben oder nicht. Dabei kann solange nichts Schlimmes passieren, wie kein Grenzfall auftritt – doch dazu später.

Zunächst ein konkretes Beispiel: Das gespeicherte Vokabular sei
(LADE SPEICHERE DATEI DATENSATZ),
die zugehörige Phonem-Liste lautet dann

(LD SBCR D DMCDC).

Jede Eingabe, also etwa

LADE DATENSAATZ KUNDEN,

wird in die Phonemform überführt, im Beispiel

LD DMCDC KMDM,

wobei die ersten beiden Worte in der Phonem-Liste der gültigen Befehle gefunden und durch die Kommandos aus der anderen Liste ersetzt werden, während KUNDEN unverändert bleibt. Im Ergebnis hat also der Tippfehler in der Eingabe keinerlei Konsequenzen.

Mit diesem Verfahren läßt sich der Bedienugskomfort des Lisp-Interpreters deutlich steigern, wobei das Vokabular ganz den jeweiligen Bedürfnissen angepaßt werden – also nicht nur Lisp-Primitive, sondern auch Namen selbstgeschriebener Funktionen beinhalten kann. Zur Ergänzung der Kommando-Listen steht die Funktion DWIMADD (*Bild 2*) zur Verfügung. Versucht man jedoch, ein Wort einzugeben, dessen Stamm bereits registriert wurde (als Abbildung des gleichen oder eines ähnlich klingenden Wortes), so wird die Annahme verweigert.

Die Grenzen

Im anderen Falle wäre bei der Rückübersetzung nicht eindeutig, welches Wort eingesetzt werden soll – womit gleich eine Grenze dieses Systems deutlich wird: Auf den Unterschied zwischen CDR und CDRR ist es nicht anwendbar, da sonst auch aus CDDR immer ein CDR

```
(defun korr (lambda (funktion)
  (putd funktion (dwim* (getd funktion)))
  (getd funktion) ))

(defun dwim (lambda ()
  (eval (print (dwim* +)))))

(defun dwim* (lambda (liste)
   ((null liste) nil)
   ((listp (car liste))
     (cons (dwim* (car liste)) (dwim* (cdr liste))))
   (cons (test (car liste)) (dwim* (cdr liste)))))

(defun test (lambda (wort stelle)
  (setq stelle (position (uebersetze wort) dwimlist*))
  ((not (null stelle))
    (nth stelle dwimlist))
  wort))
```

Bild 3. Die Funktionen DWIM und KORR stützen sich auf die in Bild 1 angegebenen Funktionen

```
$ (dwimadd '(fakultaet zerop reverse))
T

$ (defun fakultaet (lambda (x)
    ((zerip x) 1)
    (* x (fakultaet (- x 1))) ))
FAKULTAET

$ (fakultaet 5)

Undefinded Function Break: (ZERIP X)
Continue, Break, Abort, Top-level, Restart, System? A

$ (korr fakultaet)
(LAMBDA (X) ((ZEROP X) 1) (* X (FAKULTAET (- X 1))))

$ (eval ++)
120

$ (revrse '(herum falsch ist dies))

Undefined Function Break: (REVRSE (QUOTE (HERUM FALSCH IST DIES)))
Continue, Break, Abort, Top-level, Restart, System? A

$ (dwim)
(REVERSE (QUOTE (HERUM FALSCH IST DIES)))
(DIES IST FALSCH HERUM)

$ (fakultat 6)

Undefined Function Break: (FAKULTAT 6)
Continue, Break, Abort, Top-level, Restart, System? A

$ (dwim)
(FAKULTAET 6)
720
```

Bild 4. Wer sich oft vertippt, wird an DWIM und KORR viel Freude haben

wird. Außerdem lassen sich nur Tippfehler finden, die die Phonemdarstellung eines Wortes unangetastet lassen: Ein „Womputer" wird nicht erkannt. Solche Fehler fallen jedoch meist dem Benutzer schon bei der Eingabe auf. Es ist ja gerade der Sinn der Phonemsuche, nur solche Worte auszutauschen, die hinreichend ähnlich sind. Schließlich soll der Unterschied zwischen „Kaffee" und „Affe" gewahrt bleiben und nicht der eine als falsche Schreibweise des anderen interpretiert werden. Laut Duden ist ein Phonem eben die „kleinste sprachliche Einheit, die zur Unterscheidung von Wörtern dient".

Die Praxis

Beim Programmstart müssen DWIMLIST und DWIMLIST* auf Null gesetzt oder mit einer Standard-Liste initialisiert werden. Was nun noch fehlt, steht in *Bild 3*: Die Hauptfunktionen KORR und DWIM mit den untergeordneten Routinen DWIM* und TEST. Letztere bekommt als Argument ein Wort, bildet die Phonem-Darstellung und schaut, ob sie in DWIMLIST* enthalten ist. Ist das der Fall, so liefert TEST das Wort aus DWIMLIST zurück, das natürlich an der gleichen Stelle der Liste steht. Ist es nicht enthalten, so wird das Wort unverändert zurückgegeben.

Das Argument von DWIM* ist eine Liste, die korrigiert werden soll. Jedes einzelne Element wird also an TEST übergeben; aus den von dort zurückgelieferten Worten wird eine neue Liste konstruiert. Die Zerlegung in Elemente ist bei DWIM* deshalb aufwendig, weil auch geschachtelte Listen zu berücksichtigen sind. Es genügt ja nicht, etwa bei

(DIES IST (EINE (GESCHACHTELTE (LISTE))))

nur alle Listen-Elemente zu betrachten, denn das sind zwei Atome und eine Liste, die ihrerseits verschachtelt ist. Daher muß rekursiv bis auf die Ebene der einzelnen Atome zerlegt werden.

DWIM wird ohne Argument aktiviert und korrigiert stets den zuletzt eingegebenen Ausdruck, der wie gesagt an das Atom „+" gebunden ist. KORR bekommt vom Benutzer den Namen der zu korrigierenden Funktion und bindet die verbesserte Version an diesen Namen, die dann zur Kontrolle angezeigt wird. Mit (EVAL ++) kann sie dann ohne neue Eingabe ausgeführt werden. *Bild 4* zeigt einen beispielhaften Dialog mit muLisp, der die Leistungsfähigkeit von DWIM und KORR illustriert.

Ein paar Tips

Um beim Lispeln DWIM und KORR stets zur Hand zu haben, packt man sämtliche Funktionen in eine Editor-Datei und fügt noch die Zuweisungen

(SETQ DWIMLIST NIL) und
(SETQ DWIMLIST* NIL) hinzu – oder speichert statt NIL gleich eine Liste mit Standard-Vokabeln ab. Dieses Paket kann man dann jederzeit zum Editor laden.

Eine zweite Möglichkeit besteht darin, die gesamte Umgebung als SYS-Datei auf die Diskette zu schreiben (SAVE 'DWIMKORR) und dann jeweils mit LOAD zu laden. Um Namenskonflikte mit eigenen Programmen zu vermeiden, kann man z. B. an alle Funktionsnamen des DWIM-Pakets zwei $-Zeichen anhängen.

Als dritte Möglichkeit können erfahrene Benutzer des muLisp-Systems die Break-Funktion überschreiben und DWIM direkt in das Menü „Continue, Break, Abort..." aufnehmen (siehe muLisp-Manual, Seite 5-122). Das ist im Grunde einfach, jedoch zeigte sich beim Autor, daß der Break-Level vor dem Prompt „$" nicht mehr angezeigt wird, da die gleichnamige Variable nicht mehr bekannt ist.
Das vorliegende Programm zeigt, daß mit nicht einmal 50 Programmzeilen eine echte Verbesserung dieses Lisp-Systems möglich ist und demonstriert damit wieder einmal die Stärke dieser hochflexiblen Sprache. Eine Realisierung der Phonem-Suche in Pascal findet sich übrigens in [2].

muLisp-85 läuft übrigens auf MS-DOS-Rechnern. Es belegt nur ungefähr 64 kByte Speicherplatz.

Das hier beschriebene Programm kann leicht für andere Lisp-Varianten umgeschrieben werden.

Literatur

[1] *Plate, Jürgen:* Suchen – phonetisch, mc 6/1984 Seite 49.

[2] *Stede, Manfred:* Pascal-Programme zur Künstlichen Intelligenz, Heise Verlag, Hannover 1987.

Volker Senft

Arithmetische Ausdrücke analysieren

Turbo-Prolog im praktischen Einsatz

Derzeit gibt es nur wenige Programme, die in Turbo-Prolog geschrieben sind. Dieser Mangel hängt wahrscheinlich mit den Schwierigkeiten zusammen, mit denen ein Programmierer zu kämpfen hat, wenn er sich bis jetzt nur mit „gewöhnlichen" Sprachen wie z. B. Pascal beschäftigt hat. Unser Autor stellt ein kurzes Turbo-Prolog-Programm vor, das die Arbeitsweise von Prolog verdeutlicht.

Das Programm soll einen arithmetischen Ausdruck analysieren und dessen Gültigkeit feststellen. Dies wird auch als syntaktische Analyse bezeichnet. Falls der Ausdruck syntaktisch richtig ist, wird sein Zahlenwert errechnet und auf dem Bildschirm ausgegeben. Die Gültigkeit des Ausdrucks ist durch eine Grammatik festgelegt. Die Grammatik einer Sprache legt die zur Wortbildung angewandten Regeln fest. Anhand dieser Grammatik wird der Ausdruck abgearbeitet und in einem Baum abgebildet. Der Baum ist aber nicht sichtbar, sondern zeigt sich nur durch den Ablauf des Programms. Mit der Trace-Option, die durch Einfügen des Wortes Trace am Beginn der Programmdatei eingeschaltet wird, ist der Programmablauf sehr schön zu verfolgen. Die verwendeten Zahlen eines arithmetischen Ausdrucks bilden in einem Baum die Blätter und die arithmetischen Operationen stellen die Knoten des Baumes dar. Der Ausdruck 2*4+6/3 kann durch den Baum in *Bild 1* wiedergegeben werden.

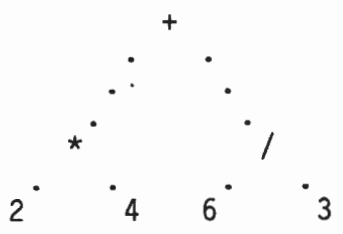


Bild 1. Der Baum zum Ausdruck 2*4+6/3

Baumstrukturen

Durch die Anordnungen im Baum ist die Vorgehensweise zur Bestimmung des arithmetischen Wertes deutlich sichtbar. Zuerst müssen die Werte für 2*4 und 6/3 errechnet werden. Nach der folgenden Addition der Teilergebnisse ist der Zahlenwert des Ausdrucks richtig bestimmt. Für die allgemeine Beschreibung solcher Ausdrücke ist eine Grammatik aufzustellen, welche die Produktion von allen möglichen Ausdrücken erlaubt. Die Grammatik, die auch später im Programm verwendet wird, ist in *Bild 2* dargestellt und beschreibt alle arithmetischen Ausdrücke, die jeder übliche Taschenrechner verarbeiten kann. Der arithmetische Ausdruck kann beliebig geklammert sein. Selbstverständlich wird bei der Berechnung des Zahlenwertes die Punkt-vor-Strich-Regel beachtet. Diese Rechenregel wird bereits durch die gegebene Grammatik berücksichtigt. Die großgeschriebenen Wörter in der Grammatik werden nicht-terminale Symbole genannt. Sie sind Platzhalter für die unterstrichenen Zeichen, die als terminale Symbole bezeichnet werden. Ein arithmetischer Ausdruck besteht nur aus terminalen Symbolen.

In der Grammatik einer natürlichen Sprache wäre der Begriff terminale Symbole ein Oberbegriff für Wörter, die in Sätzen stehen. Nicht-terminale Symbole wäre der Oberbegriff für die Satzbestandteile (Subjekt, Prädikat, Objekt). Die Analyse des arithmetischen Ausdrucks geschieht durch das Ersetzen der linken Seite mit der rechten Seite der Grammatik. Die linke Seite kann durch einen beliebigen Grammatikteil der rechten Seite ersetzt werden, wobei die Grammatikteile durch senkrechte Striche getrennt sind. In unserem Beispiel wird der Ausdruck, der zuerst einem EXPR entspricht, durch EXPR+TERM ersetzt. Nach weiteren Ersetzungen von EXPR und TERM entsteht ein Baum, wie er im *Bild 3* dargestellt ist. Die Analyse endet, wenn alle nicht-terminalen Symbole durch terminale Symbole ersetzt worden sind.

Mit dem Grammatikteil -VALUE werden auch negative Zahlen wie z.B. -823 zugelassen. Das Minuszeichen wird sonst nur als Subtraktionsoperator verwendet. Alle Zahlen, die auf eine andere Weise in einem arithmetischen Ausdruck angegeben sind (z.B 56-24; Zahl 56 und 24 sind positiv!), werden als positive ganze Zahlen betrachtet. Es ist von der gegebenen Grammatik nicht erlaubt, ein positives Vorzeichen zu verwenden. Die Zahl + 24 gibt keinen gültigen arithmetischen Ausdruck im Sinne der verwendeten Grammatik. Im unserem Beispielprogramm (*Bild 4*) wird mit ganzen Zahlen im Bereich von -32768...32767 gerechnet. Entsteht bei einer Division ein Rest, so wird das Ergebnis zur nächsten ganzen Zahl gerundet.

```
linke Seite = rechte Seite: Grammatikteil1 ! Grammatikteil2 ..

EXPR      =  EXPR + TERM  !  EXPR - TERM  !  TERM
TERM      =  TERM * FAKTOR  !  TERM / FAKTOR  !  FAKTOR
FAKTOR    =  VALUE ! - VALUE
VALUE     =  Zahlenwert  !  ( EXPR )
```

Bild 2. Grammatik für arithmetische Ausdrücke

Bevor das Programm beschrieben werden kann, müssen wir uns mit dem Umgang von Listen in Turbo-Prolog beschäftigen. Eine Liste wird gebildet durch die Aneinanderreihung eines beliebigen Datentyps. Dieser kann eine ganze Zahl (integer), eine Gleitkommazahl (real), ein strukturierter Datentyp oder wie in unserem Beispielprogramm eine Zeichenkette (string bzw. symbol) sein. Die Vereinbarung einer Liste erfolgt im Bereich der Typendeklaration, die durch das Schlüsselwort domains gekennzeichnet ist. Die Liste wird durch das Anhängen eines Sterns (*) an den Datentyp markiert. In unserem Beispielprogramm ist dies durch die Zeile Lst = Str* geschehen, wobei Str als String vereinbart ist. Eine Liste vom Typ Lst könnte wie folgt ausssehen: ["das", "ist", "eine", "Liste", "in", "Turbo-Prolog"]. Es ist natürlich auch möglich, daß eine Liste kein Element enthält und nur aus den eckigen Klammern besteht (leere Liste).
Um auf die Listenelemente zugreifen zu können, ist es möglich, die Liste in zwei Teile aufzuspalten. Die ursprüngliche Liste wird dabei in einen Kopf und in eine Restliste aufgeteilt. Der Kopf entspricht dem Datentyp aus dem die Liste aufgebaut ist. In der Restliste befinden sich alle übrigen Elemente der ursprünglichen Liste. Um die weiteren Elemente

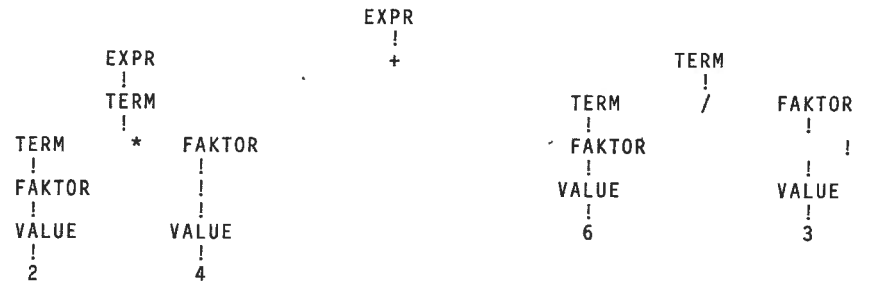


Bild 3. So sieht der Baum eines arithmetischen Ausdrucks aus, wenn die Grammatik aus Bild 2 verwendet wird

der Restliste zu erreichen, muß die Aufteilung in Kopf und Restliste noch mehrere Male vorgenommen werden. Das Aufspalten der Liste erfolgt durch den |-Operator. Vor dem |-Operator steht der Kopf der ursprünglichen Liste. Nach dem Operator folgt die Restliste. Auch der umgekehrte Vorgang ist möglich, wobei ein Element an eine Liste angehängt werden kann. Das Anhängen und Abtrennen von Kopfelementen einer Liste wird im Beispielprogramm durchgeführt, so daß kein separates Beispiel notwendig ist.

Damit die erste Zeile unserer Grammatik erfüllt werden kann, müßte die Regel EXPR sofort wieder die Regel EXPR und anschließend die Regel TERM aufrufen. Das wäre z.B. notwendig, um den Ausdruck EXPR+TERM zu analysieren. Diese Vorgehensweise ist aber nicht durchführbar, da durch den rekursiven Aufruf von EXPR eine Endlosschleife entsteht. Die gleichen Probleme entstehen bei dem Grammatikteil EXPR-TERM sowie bei der Regel TERM für die Grammatikteile TERM*FAKTOR und TERM/FAKTOR. Das Umdrehen der Regel EXPR+TERM zu TERM+EXPR ist nicht möglich, weil dadurch der arithmetische Ausdruck nicht mehr richtig berechnet werden kann. Der Ausdruck 2 – 3 + 1 hätte dann den Wert – 2 = 2 – (3 + 1)

```
/*-------------------------------------------------------------*/
/*          Typendeklaration                                   */
/*-------------------------------------------------------------*/

domains
 Str = String
 Int = Integer
 Lst = Str*

/*-------------------------------------------------------------*/
/*          string_to_list (Str,Lst,Lst)                       */
/*-------------------------------------------------------------*/
predicates
 string_to_list (Str,Lst,Lst)

clauses
 string_to_list ("",L,L).

 string_to_list (Str1,ListIn,ListOut) if
  fronttoken (Str1,Head,Str2) and
  string_to_list (Str2,[Head|ListIn],ListOut).
/*************************************************/

 /*-------------------------------------------------------------*/
 /*                  Grammatik                                  */
 /*                  =========                                  */
 /* EXPR ::= EXPR + TERM | EXPR - TERM | TERM                   */
 /* TERM ::= TERM * FAKTOR | TERM / FAKTOR | FAKTOR             */
 /* FAKTOR ::= VALUE | - VALUE                                  */
 /* VALUE ::= Integer | ( EXPR )                                */
 /*-------------------------------------------------------------*/
```

Bild 4. Das Analyse-Programm in Turbo-Prolog

```
predicates
 expr (Lst,Int,Lst)
 expr_choice (Lst,Int,Int,Lst)
 term (Lst,Int,Lst)
 term_choice (Lst,Int,Int,Lst)
 faktor (Lst,Int,Lst)
 faktor_choice (Lst,Int,Int,Lst)
 value (Lst,Int,Lst)

/*-------------------------------------------------------------*/
/*          value (Lst,Int,Lst)                                */
/*-------------------------------------------------------------*/

clauses
 value ([Value|ListOut],Val,ListOut) if
  str_int (Value,Val).

 value (ListIn,EXPR,ListOut) if
  ListIn=[")"|List1] and
  expr (List1,EXPR,List2) and
  List2=["("|ListOut].
/*************************************************/

/*-------------------------------------------------------------*/
/*          faktor (Lst,Int,Lst)                               */
/*-------------------------------------------------------------*/

clauses
 faktor_choice (["-"|List],VALUE,ValOut,List) if
  ValOut=-VALUE.

 faktor_choice (List,Val,Val,List).
 faktor (ListIn,Val,ListOut) if
  value (ListIn,VALUE,List1) and
  faktor_choice (List1,VALUE,Val,ListOut).
/*************************************************/
```

und nicht den richtigen Wert 0. Das Problem wird in dem Beispielprogramm durch Umdrehen des arithmetischen Ausdrucks gelöst. Somit wird der Ausdruck vom Ende her analysiert und jeder Grammatikteil kann von rechts abgearbeitet werden. Die Endlosschleife durch den rekursiven Aufruf von EXPR bzw. TERM ist dadurch verhindert.

Die Regeln im Detail

Die Regel string_to_list zerlegt den arithmetischen Ausdruck in eine Liste.

Gleichzeitig werden die Elemente in umgekehrter Reihenfolge in die Ergebnisliste eingetragen. In dieser Regel wird der String "2*4 + 6/3" in die Liste ["3", "/", "6", "+", "4", "*", "2"] umgewandelt.

Die Regel „rechnung" liest den arithmetischen Ausdruck von der Tastatur ein und wandelt den Ausdruck mit der Regel string_to_list in eine Liste um. Anschließend wird diese Liste der Analyse übergeben und der berechnete Zahlenwert auf dem Bildschirm ausgegeben. Die Analyse der Liste wird nur dann als richtig akzeptiert, wenn alle Elemente der Liste bearbeitet worden sind und das Analyseprogramm eine leere Liste zurück gibt.

Die Analyse des arithmetischen Ausdrucks wird durch die Regel expr aufgerufen. Diese Regel überprüft, ob die Elemente in der Eingabeliste einen TERM darstellen. Diese Vorgehensweise wurde durch das Vertauschen des arithmetischen Ausdrucks ermöglicht. Anschließend wird durch den Aufruf der Regel expr_choice versucht, die Grammatik für EXPR weiter aufzulösen. Die drei Regeln für expr_choice entsprechen den drei Möglichkeiten um die Grammatik für EXPR zu erfüllen. Am Ende jeder Regel expr_choice wird die arithmetische Operation (Addition und Subtraktion) ausgeführt.

Mit der gleichen Vorgehensweise wie bei EXPR wird die Regel für TERM ausgeführt. Zuerst wird festgestellt, ob ein FAKTOR vorliegt. Ist dies der Fall, so wird mit der Regel term_choice versucht, den restlichen Teil der Grammatik für TERM zu erfüllen. Die drei Regeln für term_choice entsprechen dabei den Möglichkeiten der Grammatik für TERM. Am Ende der Regel term_choice wird wie bei der Regel expr_choice die arithmetische Operation (Multiplikation und Division) ausgeführt.
In der Regel faktor wird die Grammatik für die Verarbeitung von negativen Zahlen realisiert. Würde die Regel faktor mit dem Grammatikteil +VALUE erweitert, so könnte das Beispielprogramm auch Zahlen mit positiven Vorzeichen verarbeiten.

Die Regel value ermittelt den Zahlenwert einer Ziffernfolge und erfüllt die Grammatik für die geklammerten Ausdrücke. Lassen Sie sich bei der Regel für die geklammerten Ausdrücke wegen der Reihenfolge der Prüfung nicht irritieren. Bei dieser Regel wird zuerst auf „Klammer" zu und am Ende der Regel auf „Klammer auf" geprüft. Dies hängt wieder mit der Umdrehung des arithmetischen Ausdrucks zusammen.

Zum Schluß sei noch auf die sehr interessanten Erweiterungsmöglichkeiten hingewiesen. Durch einfaches Erweitern der Regel für value, kann das Beispielprogramm auch vorgegebene Konstanten in die Rechnung einbeziehen. Diese Erweiterungsmöglichkeit ist im Programmlisting angegeben. Auch die Einführung neuer Operatoren wie z.B. MOD, DIV, AND, OR und XOR ist ohne größeren Aufwand möglich. Dazu müssen nur neue Regeln in die Grammatik bzw. in das Programm eingebaut werden. Auf Grund dieser Erweiterungsmöglichkeiten, macht es Spaß mit Turbo-Prolog zu arbeiten. Zum eigenen Experimentieren sind keine Grenzen gesetzt.

```
/*-----------------------------------------------------------*/
/*            term  (Lst,Int,Lst)                            */
/*        term_choice (Lst,Int,Int,Lst)                      */
/*-----------------------------------------------------------*/
clauses

 term_choice ([*|ListIn],FAKTOR,ValOut,ListOut) if
  term (ListIn,TERM,ListOut) and
  ValOut=TERM*FAKTOR.

 term_choice ([/|ListIn],FAKTOR,ValOut,ListOut) if
  term (ListIn,TERM,ListOut) and
  ValOut=TERM/FAKTOR.

 term_choice (List,Val,Val,List).

 term (ListIn,TERM,ListOut) if
  faktor (ListIn,FAKTOR,List1) and
  term_choice (List1,FAKTOR,TERM,ListOut).
/*-----------------------------------------------------------*/
/*            expr  (Lst,Int,Lst)                            */
/*        expr_choice (Lst,Int,Int,Lst)                      */
/*-----------------------------------------------------------*/

clauses
 expr_choice ([+|ListIn],TERM,ValOut,ListOut) if
  expr (ListIn,EXPR,ListOut) and
  ValOut=EXPR+TERM.

 expr_choice ([-|ListIn],TERM,ValOut,ListOut) if
  expr (ListIn,EXPR,ListOut) and
  ValOut=EXPR-TERM.

 expr_choice (ListIn,Val,Val,ListIn).

 expr (ListIn,EXPR,ListOut) if
  term (ListIn,TERM,List1) and
  expr_choice (List1,TERM,EXPR,ListOut).
```

```
/*-----------------------------------------------------------*/
/*                    rechnung                               */
/*-----------------------------------------------------------*/
predicates
 rechnung
clauses
 rechnung if
  nl and nl and
  write ("arithmetischer Ausdruck: ") and
  readln (Ausdruck) and
  string_to_list (Ausdruck,[],Liste) and ! and
  expr (Liste,Wert,[]) and
  write ("Ergebnis: ",Wert).
/*************************************************/
/*-----------------------------------------------------------*/
/*   Erweiterung fuer die Verwendung von                     */
/*                Konstanten                                 */
/*-----------------------------------------------------------*/

/*-----------------------------------------------------------*/
/*         constant (Str,Int)                                */
/*-----------------------------------------------------------*/
/*
predicates
 constant (Str,Int)

clauses
 constant ("HUNDERT",100).
 constant ("NULL",0).
 constant ("EINS",1).

 value ([Value|ListOut],Val,ListOut) if
  upper_lower (ValueUpper,Value) and
  constant (ValueUpper,Val).                              */
/*************************************************/

goal
 rechnung
```

Joachim Beer, Raúl Rojas

Coroutinen in C und Pascal

Parallele Prozesse

Um mehrere Prozesse quasi gleichzeitig ablaufen zu lassen, wird für die Programmiersprachen C und Turbo-Pascal ein Element von Modula-2 zugänglich gemacht; der Coprozeß.

Zwei der grundlegenden Begriffe in Programmiersprachen wie Modula-2 sind Prozeß und Coroutine. Ein Prozeß ist einfach ein Programm oder eine Prozedur, die parallel zu anderen Prozeduren laufen kann. Im Idealfall könnten wir uns einen Computer vorstellen, der über mehrere Prozessoren verfügt, und jeder davon führt ein unterschiedliches Programm aus; in diesem Fall sprechen wir von Prozessen, die nebeneinander laufen. Wenn der Computer nur über einen Prozessor verfügt, dann muß die Gleichzeitigkeit der Ausführung durch time-multiplexing simuliert werden. In diesem Fall haben wir keine reelle, sondern nur virtuelle Parallelität und die ausgeführten Prozeduren werden Coroutinen genannt [1].

Die meisten Betriebssysteme, die Multitasking in Monoprozessor-Computern unterstützen, arbeiten nach dem Muster der virtuellen Parallelität. Ein Scheduler oder Monitor verwaltet eine Liste von virtuellen Prozessen. Der Scheduler entscheidet in jedem Moment, welcher Prozeß aktiviert wird und wie lange. Danach gibt der aktivierte Prozeß die Kontrolle zurück an den Scheduler, der einen neuen Prozeß aus der Prozeßliste abholt und ausführt. Die virtuelle Parallelität erzeugt so die Illusion von einem Computer mit mehreren Prozessoren. Modula-2 ist eine der wenigen Programmiersprachen, die das Coroutinen-Konzept direkt unterstützt. Ein virtueller Prozeß wird mit dem Befehl „newprocess" definiert; es muß angegeben werden, welche Prozedur den Code für die Coroutine enthält, wie groß der für die Coroutine angelegte Stack ist und wo sich dieser befindet. Ein Deskriptor erlaubt später, die Coroutine jederzeit anzusprechen:

newprocess (Prozedur, Stackadresse, Stackgröße, Deskriptor)

Mit dem Befehl „transfer" wird dann die Kontrolle von einer Coroutine an eine andere übergeben. Es muß angegeben werden, welche Coroutine der Ursprung des Transfers und welche das Ziel ist:

transfer (source, destination)

In Modula-2 ist außerdem das Hauptprogramm als die Coroutine „main" vordefiniert, so daß das Starten der ersten Coroutine wie folgt aussieht:

transfer (main, Coroutine)

Der Prozessor wechselt von einer Coroutine zur anderen durch Transfer. Die Ausführung der aufgerufenen Coroutine geht an dem Punkt weiter, in dem es durch den eigenen letzten Transfer unterbrochen wurde. Wichtig zu beachten ist dabei, daß alle lokalen Daten einer Coroutine zwischen den Coroutinen-Aktivierungen erhalten bleiben.

Die Frage ist nun, was nötig ist, um die Coroutinen auch in anderen Programmiersprachen benutzen zu können. Es wäre schön, z. B. Coroutinen als einge-

```
#define new_process(x,y,v) \#define new_process(x,y,v) \
            newprocess(x,(int*)calloc(y,sizeof(int))+y,v)

int *process;                       /* globaler Prozess-Pointer */
int mn,c1,c2;                       /* Prozess-Deskriptoren     */
int co_routine1(),co_routine2();    /* Namen der Koroutinen     */

main()

{process=&mn;
 new_process(co_routine1,1000,&c1);  /* neue Koroutine */
 transfer(&c1);
 printf("back in main program\n");
 transfer(&c2);
}

co_routine1()

{printf("inside co_routine1\n");
 new_process(co_routine2,1000,&c2);  /* neue Koroutine */
 transfer(&c2);
}

co_routine2()

{printf("inside co_routine2\n");
 transfer(&mn);
 printf("second time inside co_routine2\n");
 transfer(&mn);
}
```

```
xdef newprocess
xdef transfer
xdef process

newprocess:

link      A6,#-4
movem.L   D0-D7/A0-A6,-(sp)   Rette alle Register
move.L    process,A1          Rette Stack-Pointer in der Deskriptor-
move.L    sp,(A1)             variablen des aktiven Prozesses
move.L    12(A6),A1           Pointer zum Stackbereich der Koroutine
move.L    A1,sp               Setze den neuen Stack-Pointer
move.L    process,-(sp)       Rette alten aktiven Prozess auf neuen Stack
move.L    16(A6),process      aktiver Prozess = neue Koroutine
move.L    8(A6),-(sp)         Leg Anfangsadresse auf neuen Stack
jmp       transfer            Gehe zu transfer um zurückzukommen

transfer:

link      A6,#-4
movem.L   D0-D7/A0-A6,-(sp)   Rette alle Register
move.L    process,A1          Rette Stack-Pointer in der Deskriptor-
move.L    sp,(A1)             variablen des aktiven Prozesses
move.L    8(A6),A1            Hole den Deskriptor des Transfersziels
move.L    (A1),sp             Setze den Stackpointer entsprechend
move.L    A1,process
movem.L   (sp)+,D0-D7/A0-A6   Setze alle Register zurück
unlk      A6
rts
```

Bild 1. Die C-Routinen newprocess und transfer mit den erforderlichen externen Assembleranweisungen

baute Features in Sprachen wie C oder Pascal zu haben. Die Lösung ist einfacher als es auf den ersten Blick erscheint. Die Routinen „newprocess" und „transfer" könnten in Assembler geschrieben werden und in C- oder Pascal-Programme mit dem Linker oder mittels inline-code gebunden werden. Beide Routinen müssen nur folgendes gewährleisten:

a) newprocess:
– legt den Stack der neuen Coroutine an
– initialisiert den Stack mit der Anfangsadresse der Prozedur als Rücksprungadresse des ersten Transfers

b) transfer:
– rettet die Rücksprungadresse im Stack
– rettet die Register im Stack
– neuer Stack = Stack der Coroutine
– lädt die Register aus dem Stack
– return

Dieses Muster wurde für C (in einem 68000-System unter Unix) und Turbo-Pascal (unter MS-DOS), wie in *Bild 1* und *2* gezeigt, implementiert.

Für die Implementierung des Coroutinen-Konzepts für die Sprache C wurde ein von Modula-2 leicht abweichender Weg eingeschlagen, wie dies in Bild 1 ersichtlich ist.

Die Routinen „newprocess" und „transfer" werden wie folgt aufgerufen:

newprocess (Prozedur, Stackgröße, Deskriptoradresse)

transfer (destination)

Zu beachten ist, daß in „transfer" nicht mehr angegeben werden muß, aus welcher Umgebung der Aufruf erfolgt, sondern nur noch das Ziel des Transfers. Dieses Verfahren hat einige Vorteile. Man kann z. B. ein und dieselbe Prozedur mehrfach als Coroutine initialisieren, ohne den Code replizieren zu müssen. In Modula-2 ist es nicht ohne weiteres möglich, dieselbe Coroutine in verschiedenen Aktivierungsmodi zu benutzen, da der „source"-Parameter in transfer dann nicht mehr eindeutig ist.

In der C-Implementierung gibt es eine globale Variable „process", die die Deskriptoradresse der jeweils aktiven Routine beinhaltet. Diese Variable wird im Programmteil „main" zunächst mit der Deskriptoradresse des Hauptprogramms initialisiert.

Das Anlegen des neuen Stacks für die Coroutine geschieht über den „calloc" System-call. Da der vom „calloc" zurückgelieferte Zeiger jedoch auf den Anfang (niedrigste Adresse) des neu angelegten Speicherbereichs zeigt, der Stack in C (68000/Unix-System) aber nach unten wächst, muß der Zeiger auf das Ende des neuen Speicherbereichs umgesetzt werden. Dies versteckt man am besten innerhalb eines Makros.
Auf dem 68000 werden C-Funktionsaufrufe immer mit folgendem Vorspann und Epilog versehen:

```
{$K-}
program main;

type adress  = ^integer;
     process = record
                 stack : adress;
                 size  : integer;
                 sp    : adress;
               end;

var main,p1,p2:process; stack1,stack2:adress; i,j:integer;

procedure init_stack(prozedur_ofs,stack_seg,stack_ofs:integer);
begin
  inline($8B/$4E/$08);          {mov cx,[bp+08]  ;cx=prozedur-offset     }
  inline($8B/$46/$06);          {mov ax,[bp+06]  ;ax=stack_segment       }
  inline($8B/$5E/$04);          {mov bx,[bp+04]  ;bx=stack_offset        }
  inline($8C/$D2);              {mov dx,ss       ;rette stack-segment    }
  inline($8E/$D0);              {mov ss,ax       ;ss=neue stack-seg.     }
  inline($8B/$C4);              {mov ax,sp       ;rette stack-pointer    }
  inline($8B/$E3);              {mov sp,bx       ;sp=neue stack-offs.    }
  inline($83/$EC/$08);          {sub sp,8        ;platz für transfer-arg.}
  inline($51);                  {push cx         ;push prozedur-adresse}
  inline($55);                  {push bp         ;turbo-pascal-vorspann}
  inline($89/$E1);              {mov cx,sp       ;                      }
  inline($51);                  {push cx         ;                      }
                                { rette register                        }
  inline($50/$53/$51/$52/$56/$57/$06);      {push ax,bx,cx,dx,si,di,es}
  inline($8B/$E0);              {mov sp,ax       ;sp zurücksetzen       }
  inline($8E/$D2);              {mov ss,dx       ;ss zurücksetzen       }
end;

procedure newprocess(prozedur:integer;adr:adress;stsize:integer;var
p1:process);
begin
  p1.stack:=adr;                            {rette stackadresse     }
  p1.size:=stsize;                          {rette stackgröße       }
  p1.sp:=ptr(SEG(adr^),OFS(adr^)+stsize-28);        {sp=stackadresse-14 bytes
}
                                            { (siehe init_stack)    }

  init_stack(prozedur,SEG(adr^),OFS(adr^)+stsize);{stack initialisieren    }
end;

procedure transfer(var source:adress;dest:adress);
begin
                                { rette register                        }
  inline($50/$53/$51/$52/$56/$57/$06);    { push ax,bx,cx,dx,si,di,es     }
  inline($8C/$DA);              {mov dx,ds       ;rette data-segment    }
  inline($C5/$5E/$08);          {lds bx,[bp+8]  ;lade source-stack-pointer }
  inline($89/$27);              {mov [bx],sp     ;rette aktiver stack-  }
  inline($8C/$57/$02);          {mov [bx+2],ss;parameter                }
  inline($C5/$5E/$04);          {lds bx,[bp+4]  ;lade neuer stack in ds,bx }
  inline($8C/$D8);              {mov ax,ds       ;                      }
  inline($8E/$D0);              {mov ss,ax       ; lade neuer stack-seg.}
  inline($8B/$E3);              {mov sp,bx       ;neuer stack-pointer   }
  inline($8E/$DA);              {mov ds,dx       ;data-segment rücksetzen }
                                { pop register                          }
  inline($07/$5F/$5E/$5A/$59/$5B/$58/$5D);{ pop es,di,si,dx,cx,bx,ax,bp }
end;

procedure test2;
var x:integer;
begin
writeln('in test2');
x:=1;
transfer(p2.sp,main.sp);
writeln('zweites mal in test2, x=',x,' j=',j);
transfer(p2.sp,main.sp);
end;

procedure test;
begin
transfer(p1.sp,main.sp);
j:=7;
transfer(p1.sp,main.sp);
writeln('drittes mal in test');
transfer(p1.sp,main.sp);
end;

begin
getmem(stack1,4096);            {lege stack für koroutine 1 an}
getmem(stack2,4096);            {lege stack für koroutine 2 an}

newprocess(OFS(test),stack1,4096,p1);
newprocess(OFS(test2),stack2,4096,p2);

writeln('transfer zu test');
j:=5;
transfer(main.sp,p1.sp);    writeln('zurück aus test');
transfer(main.sp,p1.sp);    writeln('transfer zu test2');
transfer(main.sp,p2.sp);    writeln('zurück aus test2');
transfer(main.sp,p1.sp);    writeln('zurück aus test');
transfer(main.sp,p2.sp);    writeln('im hauptprogramm');

halt;
end.
```

Bild 2. Newprocess und transfer in Turbo-Pascal

LINK A6,#x	x = benötigter Stackbereich
MOVEM.L Xi-Xj,-(sp)	die Register werden gerettet
(Code der Funktion)	
MOVEM.L +(sp),Xi-Xj	Rücksetzen der Register
UNLK A6	Freigeben des Stackbereichs
RTS	

Das übrige zur Implementierung von „newprocess" und „transfer" ergibt sich aus dem Listing.

Interessant ist eventuell noch das Zusammenspiel von „newprocess" und „transfer". Der erste initialisiert ja nur die neue Coroutine und springt dann wieder zurück, ohne Kontrolle an die Coroutine übergeben zu haben. Um Code zu sparen, benutzt die Coroutine „newprocess", aber auch „transfer", um zurückzuspringen.

Das Bild 2 zeigt die Definition der Prozeduren „newprocess" und „transfer" in Turbo-Pascal. Wir haben einen neuen Typ „process" als Record definiert, in dem die Anfangsadresse und die Größe des Stacks sowie der Stack-Pointer gespeichert werden. Newprocess rettet die Stackadresse und die Größe des Stacks in den Prozeßvariablen und ruft dann init_stack auf (der nur aus inline-code besteht). Diese Prozedur initialisiert einfach den neuen Stack und legt als Rücksprungadresse für den ersten Transfer die Anfangsadresse der Coroutinen-Prozedur ab.

Die Prozedur transfer rettet die Register und die Rücksprungadresse im Stack und wechselt dann die Stacks. Ein normales Return wird dann ausgeführt. Für das Verständnis des Listings ist es nur wichtig zu wissen, daß Turbo-Pascal 3.0 immer den Code der Prozeduren mit folgendem Vorspann versieht:

```
push bp
mov bp,sp
push bp
```

Am Ende einer Prozedur wird dann folgender Epilog benutzt:

```
mov sp,bp
pop bp
```

Dies ist in den Prozeduren init_stack und transfer berücksichtigt worden. Die Kommentare zu den Prozeduren erklären alles weitere. Wenn das Programm des Listings als COM-Datei compiliert wird, kann mit Debug die Ausführung des Transfers verfolgt werden.

Da Turbo-Pascal die globalen Variablen in dem Daten-Segment speichert, kann jede Coroutine auf sie zugreifen. Man muß aber aufpassen, nicht lokale Prozeduren (Prozeduren, die in anderen Prozeduren definiert sind) als Coroutinen zu definieren, da diese die lokalen Variablen der übergeordneten Prozedur nicht erreichen können (da sie in einem anderen Stack angelegt werden). Das Hauptprogramm muß außerdem mit Halt gestoppt werden.

Literatur

[1] *Wirth, N.*: Programming in Modula-2. Third, Corrected Edition, Springer-Verlag 1985.

[2] Turbo-Pascal Reference Manual. Borland International.

Impulsübertragung via V.24

Die beschriebene Problemlösung resultiert aus einer überhitzten Diskussion, in der behauptet wurde: „... mit dem Blechotto kann man ohne Investition von 1000 DM noch nicht einmal die Einschaltdauer eines Kühlschranks auswerten ...".
Gegenüber Heimcomputern, wie z. B. dem C-64, haben PCs nur genormte V.24- oder Parallelschnittstellen. Einfachste meßtechnische Aufgaben sind mit diesen Computern daher oft nur mit überproportionalem Zusatzaufwand zu lösen.
Bei dem gegebenen Problem – Einschaltdauer eines Gerätes – genügt es, dem Rechner die „EIN"- bzw. „AUS"-Meldung (die elektrisch einfach als Impuls realisierbar ist) durch Eingabe eines beliebigen ASCII-Zeichens über die V.24-Schnittstelle mitzuteilen.
Dieses Zeichen kann für beide Zustände gleich sein.
Zusätzlich muß man wissen, daß die meisten V.24-Schnittstellen zwar mit ±15-V-Pegel spezifiert sind, aber trotzdem mit 0 V als „Space"-Pegel und >5 V als „Mark"-Pegel arbeiten.
Aus der ASCII-Codetabelle ist zu ersehen, daß einige Zeichen bei Berücksichtigung des Startbits einen „Impuls" darstellen. Die Länge dieses Impulses beträgt bei 300 Baud Übertragungsgeschwindigkeit für das Zeichen „x" plus Startbit ca. 12 ms.
Nun, der Rest ist einfach: Triggerschaltung plus Impulsgenerator für 12 ms (*Bild 1*) ergibt ASCII-Zeichen „x" für eine serielle Schnittstelle mit 300 Baud, N,7,1.

Mir ist natürlich bekannt, daß es komfortable ICs gibt, mit denen die Aufgabe eleganter und mit weniger Denkarbeit gelöst werden kann, aber: ... wer kennt die Namen, nennt die Quellen ...?
Was beim ersten Hinsehen wie die „Vergewaltigung" einer V.24-Schnittstelle aussieht, ist bei genauer Betrachtung aber absolut korrekt. Dies zeigt sich auch in der Reproduktionsmöglichkeit von Aufwand und Ergebnis. *E. M. Heide*

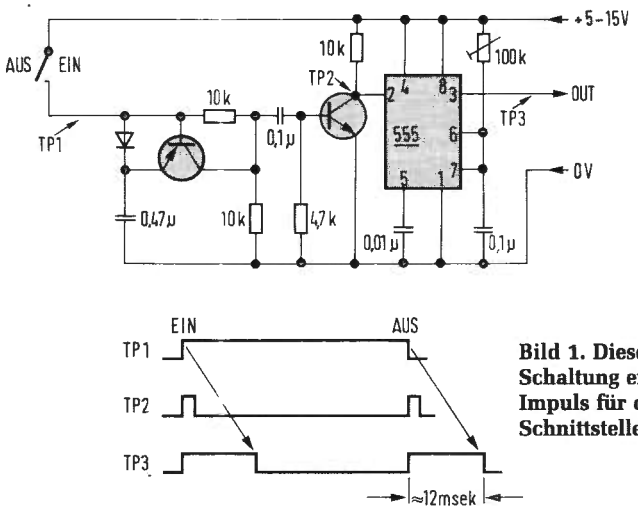


Bild 1. Diese kleine Schaltung erzeugt einen Impuls für die serielle Schnittstelle

Hendrik Meyer

Hardcopy von CGA-Farbgrafiken

Farben durch unterschiedliche Graustufen darstellen

Mit dem Graphics-Treiber läßt sich über die PrtSc-Taste schnell eine Hardcopy des Grafik-Bildschirms erzeugen. Dabei wird das Bild allerdings auf eine DIN-A4-Seite vergrößert, und die Konturen des Bildes werden sehr grob. Eine Turbo-Pascal-Prozedur schafft hier Abhilfe.

Die Turbo-Pascal-Prozedur kann Bilder der CGA-Karte punktweise ausdrucken, spiegeln und invertieren. Mit einem 24-Nadel-Drucker können Grafiken mit voller Auflösung vergrößert oder verzerrt ausgedruckt werden. Farbbilder sind mit deutlichem Kontrast ausdruckbar, da jeder der vier Farben eine unterschiedlich intensive Schraffur zugeordnet wird. Bilder können abgespeichert und später wieder geladen werden, um sie erneut auszudrucken. Das lohnt sich besonders bei Experimenten mit dem Apfelmännchen und Julia-Mengen, von denen einzelne Bilder den Rechner stundenlang beschäftigen können.

Die technischen Daten der Prozedur

Die Prozedur ist in Turbo-Pascal 3.0 auf einem IBM-PC bzw. einem kompatiblen Computer mit Color-Grafik-Karte (CGA) erstellt und getestet worden. Sie wird als Include-Datei eingebunden und vergrößert den Code um ungefähr 6 KByte. Um die Möglichkeiten der Vergrößerung von Bildern und des Kontrastdruckes bei Farbgrafiken ausnutzen zu können, benötigt man einen Matrix-Drucker mit 24

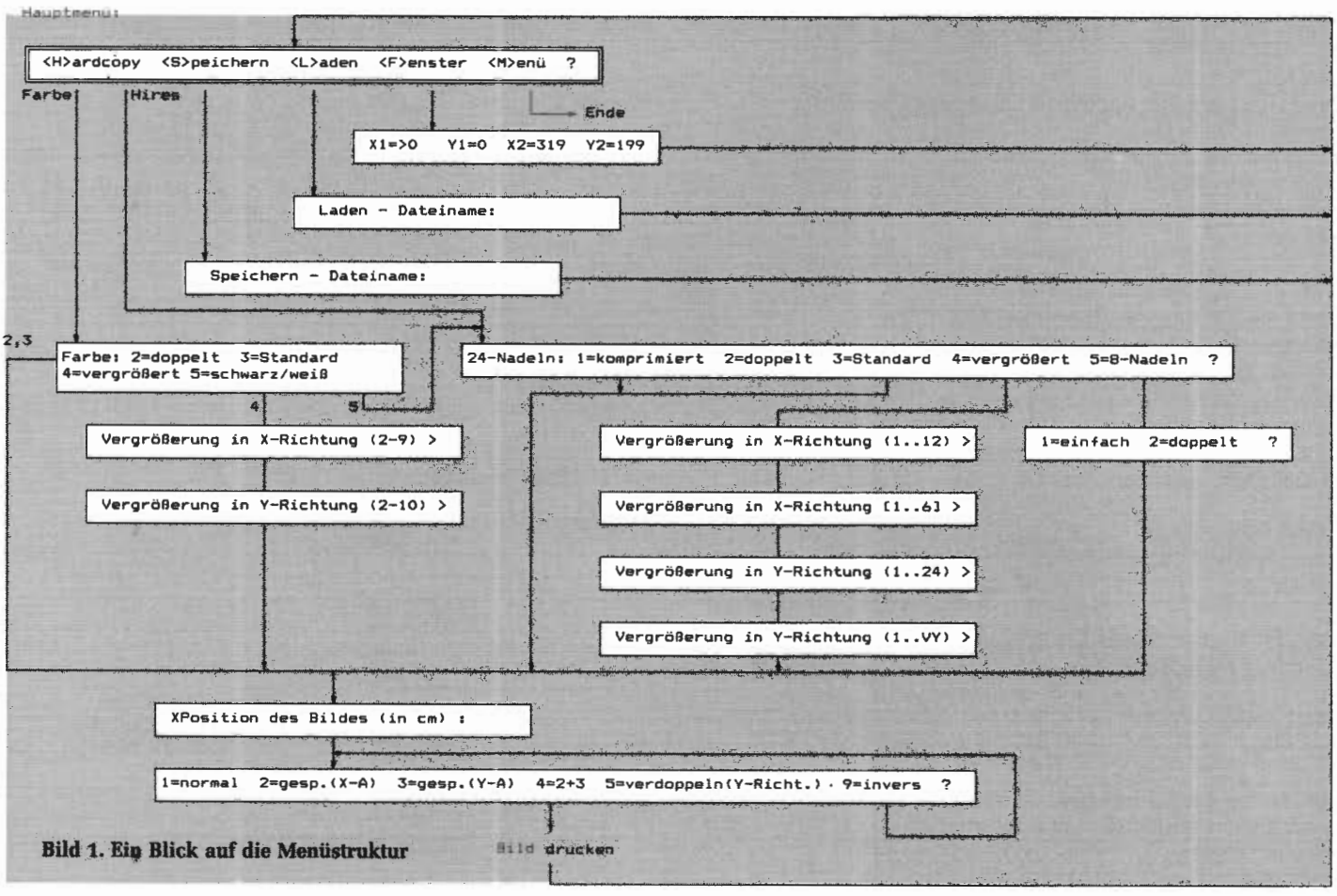


Bild 1. Ein Blick auf die Menüstruktur

Nadeln. Mit einem 9-Nadel-Drucker lassen sich Bilder nur spiegeln und invertieren.

Bedienung der Prozedur

Der Dialog zwischen Benutzer und Hardcopy-Routine geschieht über die unterste (bei Hires 640 x 200 Punkte) bzw. die untersten zwei Bildschirmzeilen (im Farb-Modus 320 x 200 Punkte). Der gesamte Bildschirminhalt wird von der Prozedur gerettet, bevor sie irgendwelche Veränderungen vornimmt, so daß auch die untersten Zeilen ausdruckbar sind. Vor Veränderungen durch DOS-Fehlermeldungen (etwa wegen Papierende oder nicht geschlossener Laufwerksklappe) ist das Bild gesichert. Nach Beendigung der Hardcopy-Prozedur wird der Bildschirminhalt wiederhergestellt.
Das Fenster, das gedruckt oder abgespeichert werden soll, wird durch ein Rechteck eingerahmt, das nicht mitgedruckt wird.

Ein reichhaltiges Menü

Das Programm ist menügeführt. Der Text, der jeweils in der untersten Bildschirmzeile erscheint, ist immer umrahmt. Wenn am Ende der Menüzeile ein Fragezeichen steht, kann mit der Escape-Taste abgebrochen werden. In der Hauptmenüzeile hat das die gleiche Wirkung wie M = Menü. Von jedem anderen Punkt aus kommt man so in die Hauptmenüzeile zurück. Wenn das '>'-Zeichen als Prompt erscheint, wird die Eingabe einer Integer-Zahl gefordert. *Bild 1* gibt einen Überblick über das Menüsystem.

S: Speichern des Bildes

Das momentane Bildfenster wird auf Diskette abgespeichert. Dazu fragt das Programm nach dem Dateinamen, unter dem das Bild abgelegt werden soll. Dem Dateinamen dürfen Laufwerks- und Pfadbezeichnung vorangestellt sein. Bei einer leeren Eingabe erscheint wieder die Hauptmenüzeile. Tritt ein Fehler auf, so fragt die Prozedur erneut nach dem Dateinamen. Es wird jedoch nicht erkannt, wenn unter dem angegebenen Namen schon eine Datei existiert. Eine bereits bestehende Datei wird überschrieben!
Auf Diskette belegt die Datei immer 16400 Bytes unabhängig von der Größe des Bildfensters. Die Dateigröße ist abhängig vom Typ der Puffer-Variablen, die für den ganzen Bildschirm dimensioniert ist, und nicht von deren Inhalt.

L: Laden einer Bilddatei

Für die Angabe des Dateinamens gilt das gleiche, wie beim Abspeichern. Die Prozedur fragt so lange nach, bis die angegebene Datei gefunden wurde. Beim Laden wird das momentan angezeigte Bild überschrieben. Das geladene Bild wird in die linke obere Ecke des Bildschirms positioniert, die Fenster-Werte und der Bildschirmmodus werden entsprechend angepaßt.

F: Verändern des Bildschirmfensters

Die Größe des zu druckenden oder abzuspeichernden Bildfensters ist einstellbar. X1/Y1 ist die linke obere und X2/Y2 die rechte untere Ecke dieses Fensters.

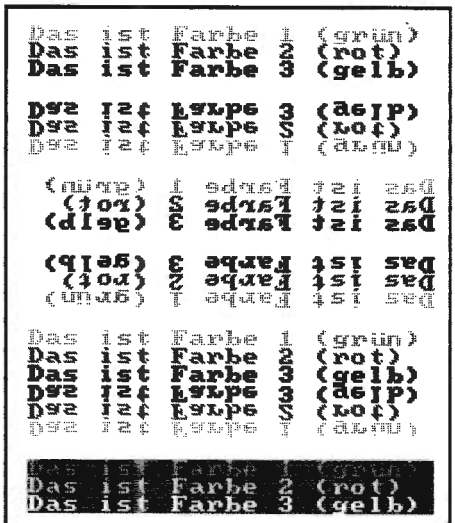


Bild 3. Farben werden als unterschiedliche Schraffur wiedergegeben

Drückt man bei einer Zahl einfach <Return>, so wird der alte Wert übernommen; sonst muß die neue Zahl vollständig eingegeben werden, auch wenn vielleicht noch Ziffern auf dem Bildschirm stehen, die man übernehmen wollte.

H: Hardcopy

Das eingerahmte Bild wird gedruckt. Zuvor muß aber noch eine Reihe weiterer Eingaben gemacht werden. Als erstes ist die Bildvergrößerung zu definieren; sie ist bei Farb- und Hires-Bildern unterschiedlich. Mit einem 24-Nadel-Drucker läßt sich das Bild stufenweise vergrößert ausdrucken. Die Werte 1 bis 3 sind voreingestellte Größen. Mit 4 ist die Vergrößerung in X- und Y-Richtung jeweils einzeln einstellbar. Das Bild läßt sich

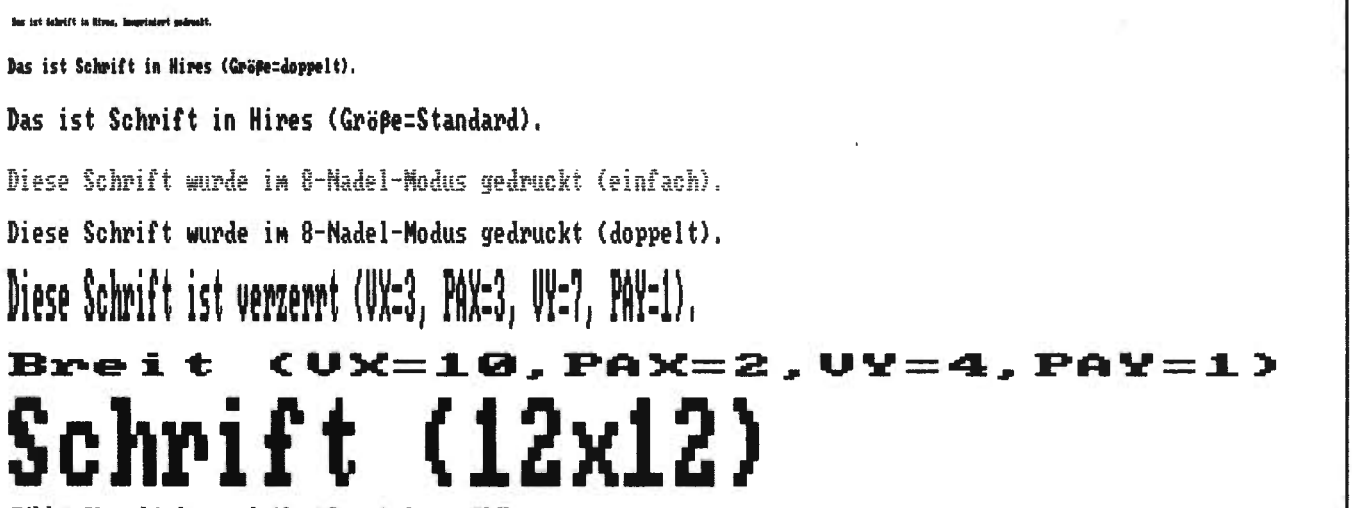


Bild 2. Verschiedene Schriftgrößen sind auswählbar

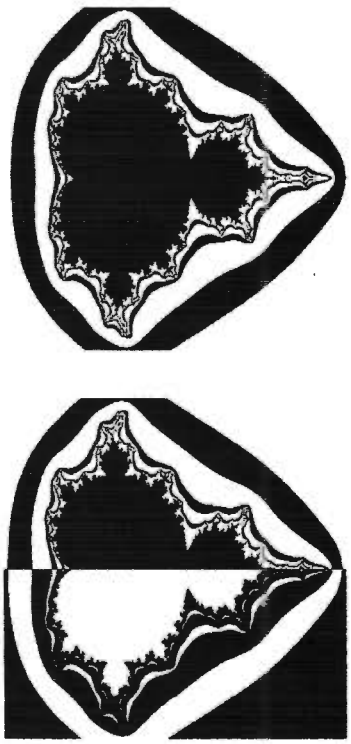


Bild 4. Das Apfelmännchen teilweise invertiert und normal ausgedruckt

Spruch des Monats

„Desktop publishing ist sicher eine neue Stufe der Freiheit. Wir können mehr Informationen leichter bearbeiten und leichter veröffentlichen als je zuvor. Aber der Preis ist: Mit dieser totalen Manipulierbarkeit kommt auch die totale Verantwortung."

Stanley J. Wszola, Redakteur bei Byte

auch verzerrt ausdrucken (*Bild 2*). Bei einem Hires-Bild muß außer der Vergrößerung auch der Abstand der Druckerpunkte eingegeben werden. In der Regel wird der mögliche Bereich in Klammern angezeigt. Beim Punktabstand in X-Richtung ist es eine Empfehlung, die meist zwischen 2 und 4 liegt. Bei einem größeren Wert wird das Bild grobkörnig. Da der horizontale Punktabstand falls möglich durch Benutzung des Druckersteuercodes für die entsprechende Druckdichte erreicht wird, verringert sich durch die Wahl eines sinnvollen (vorgeschlagenen) Punktabstandes erheblich die Anzahl der Bytes, die zum Drucker übertragen werden müssen. Die Druckzeit kann so deutlich verringert werden. Sonst wird mit 1/360 Zoll Punktabstand gedruckt und es müssen eine entsprechende Menge Nullen als Füllbytes übertragen werden.

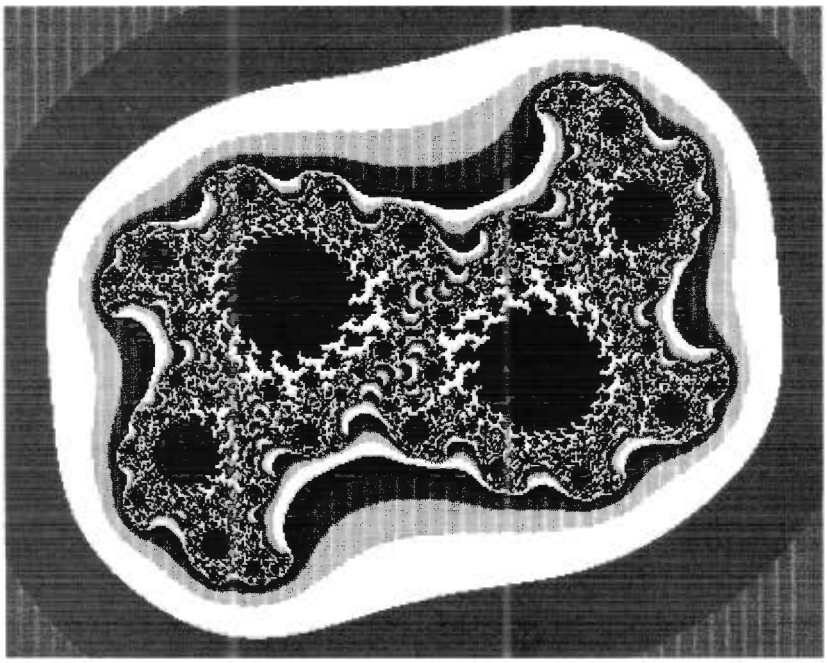


Bild 5. So sieht eine Julia-Menge aus – berechnet wurde nur die obere Hälfte und dann doppelt gespiegelt

Dieses ist z. B. auch bei 5- oder 7-facher oder einer anderen krummen Vergrößerung der Fall.
Soll ein Farbbild wie ein Hires-Bild ausgedruckt werden, so muß bei der Vergrößerung 5 eingegeben werden, um in das Vergrößerungs-Menü für Hires-Bilder zu gelangen. Das Farbbild wird nun Bit für Bit interpretiert und es besteht kein sichtbarer Unterschied mehr zwischen den Farben 1 und 2. Vergrößert man das Bild stärker als 2 bis 3mal, so werden diese Farben unangenehm auseinandergerissen.
Falls ein 9-Nadel-Drucker angeschlossen ist oder ein Ausdruck schnell gebraucht wird, so muß im Hires-Menü 5 für 8-Nadel-Druck eingegeben werden. Das Bild ist genau so groß wie beim 24-Nadel-Druck mit dreifacher Vergrößerung. Hier muß man angeben, ob das Bild einfach oder doppelt gedruckt werden soll. Dies bezieht sich nicht auf die Größe, sondern auf die Druckdichte. Bei doppelter Druckdichte wird jeder Punkt vertikal um einen Drittelpunkt (bei 24-Nadel-Druckern 1/180 Zoll, bei 9-Nadel-Druckern 1/216 Zoll) versetzt noch einmal gedruckt. Bei einem 9-Nadel-Drukker wird das Bild vertikal etwas zusammengedrückt.

Linker Rand

Als nächstes muß man eingeben, um wieviele cm das Bild vom linken Papierrand entfernt gedruckt werden soll. Beim 8-Nadel-Druckmodus ist das Maximum 5 cm. Es wird nicht überprüft, ob das Bild bei der eingegebenen Vergrößerung evt. den Papierrand überschreitet.

Spiegeln und Invertieren des Bildes

Als letztes muß die Ausrichtung des Bildes festgelegt werden. Die Ergebnisse der verschiedenen Richtungen sind in

Bild 3 zu sehen. Das Apfelmännchen in *Bild 4* ist mit 5 = Verdoppeln erzeugt worden. Berechnet wurde nur die obere Hälfte. Damit läßt sich ein symmetrisches Bild mit doppelter Auflösung bzw. in der halben Rechenzeit erzeugen. Da vor und nach dem Drucken des Bildes kein zusätzlicher Zeilenvorschub durchgeführt wird, kann man das Bild auch anders zusammensetzen. Die Julia-Mengen z. B. sind punktsymmetrisch. Ich habe daher hier nur die obere Hälfte berechnet und diese zuerst mit 1 = normal und direkt danach noch einmal bei derselben Vergrößerung mit der Richtung 4 ausgedruckt (*Bild 5*).
Möchte man das Bild invertieren, so muß man 9 eingeben.
Man gelangt dann wieder in diese Menüzeile, um auch noch die Ausrichtung des Bildes angeben zu können.

Einbindung in eigene Programme

Der Programmtext aus *Bild 7* wird am günstigsten in eine Include-Datei gepackt (Hardcopy.Inc). Im Hauptprogramm muß man diese nach der zu Turbo-Pascal gehörenden Graph.P-Datei einbinden, aus der die Befehle GetPic und PutPic benötigt werden.
Im Programm wird der Aufruf der Prozedur Hardcopy nach dem Programmstück, das die Grafik produziert, eingesetzt.
Das Prinzip zeigt das kleine Programm in *Bild 6*. Gibt man dieses so ein, so kann man damit auf Diskette gespeicherte Bilder laden und erneut ausdrucken.
Die Parameter von Hardcopy geben ein Default-Fenster an (linke obere Ecke, rechte untere Ecke), das man dann natürlich noch ändern kann.
Außer dem Prozedurnamen Hardcopy werden keine weiteren Namen global definiert, so daß sich die Prozedur schnell und ohne Namenskonflikte in grafikerzeugende Programme einfügen läßt.

Die Prozedur (*Bild 7*) ist in sechs Teile gegliedert. Wesentlich ist die zentrale Prozedur (DruckeFenster: Zeilen 70...187) zur Umwandlung des Bildschirminhalts in die Druckerdaten.

Wie wird der Bildschirminhalt in Druckerdaten umgewandelt?

Das zu druckende Bildschirmfenster wird mit dem Befehl GetPic (Buffer, X1, Y1, X2, Y2) in die Variable Buffer (ein Array of Byte) eingelesen (Zeile 212). In dem ersten Byte dieses Feldes steht der Bildschirmmodus:

```
Program Beispiel;

(*$I a:Graph.P*)        (* Laufwerk evt. anpassen! *)
(*$I Hardcopy.Inc*)

Begin
  Hires;                (* Grafikmodus einschalten *)

  (* Hierhin kommt das Programmstück zum
                              Erstellen der Grafik *)

  Hardcopy(0,0,619,199); (* Den gesamten Bild-
                         schirm als Fenster annehmen *)

End.
```

Bild 6. So wird die Prozedur in ein Programm eingebunden

1 = Hires, 2 = Farbmodus (halbe Auflösung, 4 Farben). Im dritten und vierten Byte steht, wie groß das im Puffer gespeicherte Fenster in X-Richtung und im fünften und sechsten Byte, wie groß es in Y-Richtung ist (zuerst Low-Byte, dann High-Byte). Nun folgen die Daten des Bildes. Diese sind Zeile für Zeile gespeichert, immer acht nebeneinanderliegende Punkte in einem Byte bei Hires bzw. vier Punkte bei Farbe (zwei Bit pro Punkt = vier Farben). Der Drucker möchte immer übereinanderliegende Punkte zu einem Byte zusammengefaßt haben, wenn das Bild nicht quer gedruckt werden soll. Beim 8-Nadel-Druck werden in der Prozedur DruckeFenster immer acht Bildschirmzeilen auf einmal zum Drucker geschickt; beim 24-Nadel-Druck sind es je nach Vergrößerung mehr oder weniger. In der Variablen DBuf wird immer eine Zeile für den Drucker aufgebaut (Zeile 122-162).
Wenn das Bild vergrößert werden soll, so kann man einmal einfach den Punktabstand verändern, wie es durch die Wahl des entsprechenden Druckerbefehls möglich ist. Bei starker Vergrößerung erscheint das Bild löchrig. Zur Abhilfe kann ein Bildschirmpunkt in mehrere Druckerpunkte umgesetzt werden. Dazu muß man im Hires-Modus den Punktabstand angeben.
Im Farbmodus wird dieser über eine Farbtabelle gesteuert, um unterschiedliche Graustufen zu erzeugen. Die Größe dieser Farbtabelle (ab Zeile 88) bestimmt die maximale Vergrößerung im Farbmodus (hier 9 x 10). In der Prozedur ist nur für die Farben 1 bis 3 ein Bitmuster angelegt, da für die Farbe 0 das Papier weiß bleiben soll. Da die Variable DBuf zuerst gelöscht wird, müssen nicht noch einmal Nullen eingeschoben werden.

Die Parameter von DruckeFenster

Wenn Sie von Ihrem Programm aus direkt den Grafikbildschirm ausdrucken möchten, ohne Angaben zum Bildformat zu machen, dann können Sie diese Unter-Prozedur auch herauslösen und einzeln in Ihr Programm einbauen. Dazu müssen Sie die Variable Buffer entsprechend definieren und mit dem zu drukkenden Bildfenster laden und die Parameter richtig übergeben. Dabei ist besonders der Aufbau von TabulatorStr und dessen Zusammenhang mit VergrX und VergrXSchritt zu beachten: In TabulatorStr steht zuerst der Steuercode, der den linken Rand bewirkt. Dieser wird hier im 8-Nadel-Druck durch weiße Grafik erzeugt. Daher ist der Rand durch die maximale Stringlänge auf 5 cm begrenzt, wenn wie in der abgedruckten Prozedur der Steuercode Esc L benutzt wird. Es folgt der Anfang des Grafik-Befehls. Die zwei Bytes, die die Anzahl der im Bit-Muster auszudruckenden Bytes übergeben, werden in DruckeFenster angehängt (Zeile 100).
Über die Wahl des Grafikbefehls wird die Vergrößerung in X-Richtung miteingestellt. Die verwendeten Befehle stehen in dem Feld DruckBefehle (Zeile 204). Die Reihenfolge lautet: 1/360 Zoll Horizontalabstand (kleinste Größe), 1/180 Zoll (doppelte Größe), 1/120 Zoll (Standard = dreifache Größe), 1/90 Zoll, (1/72 Zoll für fünffache Größe gibt es oft nicht), 1/60 Zoll (sechsfache Größe).

Der entsprechende Code wird in der Prozedur Drucke über das Feld DruckModus, dessen erster Index die Vergrößerung und der zweite den Punktabstand in X-Richtung angibt, ausgewählt (Zeile 274-283). Der Parameter VergrX gibt die Vervielfachung eines Bildschirmpunktes an. Um die reale Vergrößerung zu erhalten, muß man diesen noch mit der des Grafikbefehls multiplizieren.

Druckeranpassung

Für alle Druckersteuercodes sind Konstanten vorgesehen, die sich ändern las-

Bild 3 zu sehen. Das Apfelmännchen in *Bild 4* ist mit 5 = Verdoppeln erzeugt worden. Berechnet wurde nur die obere Hälfte. Damit läßt sich ein symmetrisches Bild mit doppelter Auflösung bzw. in der halben Rechenzeit erzeugen. Da vor und nach dem Drucken des Bildes kein zusätzlicher Zeilenvorschub durchgeführt wird, kann man das Bild auch anders zusammensetzen. Die Julia-Mengen z. B. sind punktsymmetrisch. Ich habe daher hier nur die obere Hälfte berechnet und diese zuerst mit 1 = normal und direkt danach noch einmal bei derselben Vergrößerung mit der Richtung 4 ausgedruckt (*Bild 5*).
Möchte man das Bild invertieren, so muß man 9 eingeben.
Man gelangt dann wieder in diese Menüzeile, um auch noch die Ausrichtung des Bildes angeben zu können.

Einbindung in eigene Programme

Der Programmtext aus *Bild 7* wird am günstigsten in eine Include-Datei gepackt (Hardcopy.Inc). Im Hauptprogramm muß man diese nach der zu Turbo-Pascal gehörenden Graph.P-Datei einbinden, aus der die Befehle GetPic und PutPic benötigt werden.
Im Programm wird der Aufruf der Prozedur Hardcopy nach dem Programmstück, das die Grafik produziert, eingesetzt.
Das Prinzip zeigt das kleine Programm in *Bild 6*. Gibt man dieses so ein, so kann man damit auf Diskette gespeicherte Bilder laden und erneut ausdrucken.
Die Parameter von Hardcopy geben ein Default-Fenster an (linke obere Ecke, rechte untere Ecke), das man dann natürlich noch ändern kann.
Außer dem Prozedurnamen Hardcopy werden keine weiteren Namen global definiert, so daß sich die Prozedur schnell und ohne Namenskonflikte in grafikerzeugende Programme einfügen läßt.

Die Prozedur (*Bild 7*) ist in sechs Teile gegliedert. Wesentlich ist die zentrale Prozedur (DruckeFenster: Zeilen 70...187) zur Umwandlung des Bildschirminhalts in die Druckerdaten.

Wie wird der Bildschirminhalt in Druckerdaten umgewandelt?

Das zu druckende Bildschirmfenster wird mit dem Befehl GetPic (Buffer, X1, Y1, X2, Y2) in die Variable Buffer (ein Array of Byte) eingelesen (Zeile 212). In dem ersten Byte dieses Feldes steht der Bildschirmmodus:

```
Program Beispiel;

(*$I a:Graph.P*)        (* Laufwerk evt. anpassen! *)
(*$I Hardcopy.Inc*)

Begin
  Hires;                (* Grafikmodus einschalten *)

  (* Hierhin kommt das Programmstück zum
                              Erstellen der Grafik *)

  Hardcopy(0,0,619,199); (* Den gesamten Bild-
                         schirm als Fenster annehmen *)

End.
```

Bild 6. So wird die Prozedur in ein Programm eingebunden

1 = Hires, 2 = Farbmodus (halbe Auflösung, 4 Farben). Im dritten und vierten Byte steht, wie groß das im Puffer gespeicherte Fenster in X-Richtung und im fünften und sechsten Byte, wie groß es in Y-Richtung ist (zuerst Low-Byte, dann High-Byte). Nun folgen die Daten des Bildes. Diese sind Zeile für Zeile gespeichert, immer acht nebeneinanderliegende Punkte in einem Byte bei Hires bzw. vier Punkte bei Farbe (zwei Bit pro Punkt = vier Farben). Der Drucker möchte immer übereinanderliegende Punkte zu einem Byte zusammengefaßt haben, wenn das Bild nicht quer gedruckt werden soll. Beim 8-Nadel-Druck werden in der Prozedur DruckeFenster immer acht Bildschirmzeilen auf einmal zum Drucker geschickt; beim 24-Nadel-Druck sind es je nach Vergrößerung mehr oder weniger. In der Variablen DBuf wird immer eine Zeile für den Drucker aufgebaut (Zeile 122-162).
Wenn das Bild vergrößert werden soll, so kann man einmal einfach den Punktabstand verändern, wie es durch die Wahl des entsprechenden Druckerbefehls möglich ist. Bei starker Vergrößerung erscheint das Bild löchrig. Zur Abhilfe kann ein Bildschirmpunkt in mehrere Druckerpunkte umgesetzt werden. Dazu muß man im Hires-Modus den Punktabstand angeben.
Im Farbmodus wird dieser über eine Farbtabelle gesteuert, um unterschiedliche Graustufen zu erzeugen. Die Größe dieser Farbtabelle (ab Zeile 88) bestimmt die maximale Vergrößerung im Farbmodus (hier 9 x 10). In der Prozedur ist nur für die Farben 1 bis 3 ein Bitmuster angelegt, da für die Farbe 0 das Papier weiß bleiben soll. Da die Variable DBuf zuerst gelöscht wird, müssen nicht noch einmal Nullen eingeschoben werden.

Die Parameter von DruckeFenster

Wenn Sie von Ihrem Programm aus direkt den Grafikbildschirm ausdrucken möchten, ohne Angaben zum Bildformat zu machen, dann können Sie diese Unter-Prozedur auch herauslösen und einzeln in Ihr Programm einbauen. Dazu müssen Sie die Variable Buffer entsprechend definieren und mit dem zu drukkenden Bildfenster laden und die Parameter richtig übergeben. Dabei ist besonders der Aufbau von TabulatorStr und dessen Zusammenhang mit VergrX und VergrXSchritt zu beachten: In TabulatorStr steht zuerst der Steuercode, der den linken Rand bewirkt. Dieser wird hier im 8-Nadel-Druck durch weiße Grafik erzeugt. Daher ist der Rand durch die maximale Stringlänge auf 5 cm begrenzt, wenn wie in der abgedruckten Prozedur der Steuercode Esc L benutzt wird. Es folgt der Anfang des Grafik-Befehls. Die zwei Bytes, die die Anzahl der im Bit-Muster auszudruckenden Bytes übergeben, werden in DruckeFenster angehängt (Zeile 100).
Über die Wahl des Grafikbefehls wird die Vergrößerung in X-Richtung miteingestellt. Die verwendeten Befehle stehen in dem Feld DruckBefehle (Zeile 204). Die Reihenfolge lautet: 1/360 Zoll Horizontalabstand (kleinste Größe), 1/180 Zoll (doppelte Größe), 1/120 Zoll (Standard = dreifache Größe), 1/90 Zoll, (1/72 Zoll für fünffache Größe gibt es oft nicht), 1/60 Zoll (sechsfache Größe).

Der entsprechende Code wird in der Prozedur Drucke über das Feld DruckModus, dessen erster Index die Vergrößerung und der zweite den Punktabstand in X-Richtung angibt, ausgewählt (Zeile 274-283). Der Parameter VergrX gibt die Vervielfachung eines Bildschirmpunktes an. Um die reale Vergrößerung zu erhalten, muß man diesen noch mit der des Grafikbefehls multiplizieren.

Druckeranpassung

Für alle Druckersteuercodes sind Konstanten vorgesehen, die sich ändern las-

sen. Getestet wurde die Prozedur mit den 9-Nadel-Druckern Star-NL10 mit IBM-Modul, Brother M-1509 und IBM-Proprinter und mit dem 24-Nadel-Drucker NEC-P6. Wegen der weitestgehenden Standardisierung der Grundbefehle dürfte sie mit den meisten anderen gängigen Druckern arbeiten. Der NEC-P6 hat auch die 8-Nadel-Grafik-Befehle wie die oben genannten 9-Nadel-Drucker. Es wird dann eben nur jede dritte Nadel benutzt. Beim 8-Nadel-Druck wird nur der Grafikbefehl für 1/120 Zoll Druckdichte (Zeile 205: Esc L) und der Befehl zum Zeilenvorschub (Zeile 95: Esc J) benutzt.

Die Vergrößerung von Bildern wird von der Dialog-Prozedur Drucke nur für den 24-Nadel-Druck unterstützt. Die Prozedur DruckeFenster läßt das prinzipiell aber auch für den 8-Nadel-Druck zu. In Y-Richtung muß die Vergrößerung eine 2er-Potenz sein (1,2,4,8), da die Einstellung der Konstanten vor den Schleifen beim 8-Nadel-Druck nicht variabel durchgeführt wird. Die Grundgröße ist beim 8-Nadel-Druck dreimal größer als beim 24-Nadel-Druck.

```
(*****************************************************************************************************)
(*       Modul zum Drucken, Speichern und Laden von Grafikbildern
                ═══════════════════════════════════
          (C) Hendrik Meyer

V2.0  11. 8.1986  Spiegeln
V3.0  30.12.1986  Anpassung auf 24-Nadeldrucker, Vergrößerung von Bildern
V3.01 15. 2.1987  Anpassung der Randüberschreitung bei 8-Nadel-Druck
V3.1  28. 2.1987  Invertierung des Bildes
═══════════════════════════════════════════════════════════════════════════════════════════════════*)

Procedure Hardcopy (X1, Y1, X2, Y2 :Integer);   (*  Die Parameter X1, Y1, X2, Y2 sind Default-Werte für das
                                                    berücksichtigte Fenster                          *)

Type Puffer = Array[0..16399] of Byte; (* Puffertyp für GetPic                                       *)
     String255 = String[255];

Var Buf1,                              (* Zwischenspeicher für das ganze Bild während der Menüführung  *)
    Buffer          :Puffer;           (* Speicher für das zu druckende Bildfenster                    *)
    Grafmode        :Byte;             (* 1 = Hires (640x200), 2 = GrafColorMode (320x200)             *)
    ch              :Char;             (* Menü-Eingabe-Variable                                        *)

Procedure Rechteck(X1, Y1, X2, Y2 :Integer);
  Var col :Integer;
  Begin
  case GrafMode of 1: col := 1; 2: col := 3; End; (* Farbe des Rechteckes einstellen: maximale Intensität *)
  draw(X1,Y1, X2,Y1, col); draw(X2,Y1, X2,Y2, col); draw(X2,Y2, X1,Y2, col); draw(X1,Y2, X1,Y1, col);
  End;

(*------ Ein-/Ausgabe-Routinen ------------------------------------------------------------------------*)

Var Key             :Char;             (* Diese Variablen werden nur von InKey gesetzt                 *)
    FunctionKey, Esc :Boolean;
Function InKey :Char;                  (* holt ein Zeichen von der Tastatur; wenn Escape, dann Esc = true  *)
  Begin                                (* bei Funktions/Cursortaste ist FunctionKey = true             *)
  FunctionKey := false;   Esc := false;
  read(kbd,Key);
  if (Key=#27) and KeyPressed then Begin read(kbd,Key); FunctionKey := true End;
  if Key=#27 then Esc := true;         (* Escape-Taste extra registrieren                              *)
  InKey := Key;                        (* Der Tastencode steht zusätzlich in Key                       *)
  End;  (* Funktion InKey *)

Type String80 = String[80];

Procedure write25(Zeile :String80);    (* Druckt <Zeile> in der untersten Bildschirmzeile;             *)
  Begin                                (* diese wird zuerst gelöscht.                                  *)
  case GrafMode of
    1: Begin                           (* Hires (80 Zeichen/Zeile)                                     *)
       gotoxy(1,25);                   (* unterste Zeile zuerst löschen                                *)
       write('                                                                ');
       gotoxy(1,25); write(Zeile);     (* und dann <Zeile> anzeigen                                    *)
       End;
    2: Begin                           (* Halbe Auflösung; der Text muß in zwei Zeilen aufgespalten werden *)
       GotoXY(1,24); write('                                                ');
       GotoXY(1,25); write('                                                ');
       if length(Zeile)>39 then        (* Eigentlich nur 79 Zeichen möglich, da sonst gescrollt wird.  *)
          Begin GotoXY(1,24); write(copy(Zeile,1,40));  GotoXY(1,25); write(copy(Zeile,41,79)); End
       else                            (* Wenn <Zeile> kurz genug ist,dann nur die unterste Zeile benutzen.*)
          Begin GotoXY(1,25); write(Zeile); End;
       End;
    End;
  End;  (* Prozedur write25 *)

Procedure ReadXY(X,Y :Integer; Var a :Integer); (* liest eine Integerzahl an einer bestimmten Position  *)
  Begin
  GotoXY(X,Y); write('>'); read(a);    (* Position anzeigen, wo die Eingabe verlangt wird              *)
  GotoXY(X,Y); write(' ');             (* Prompt-Symbol wieder löschen                                 *)
  End;

(*------ Hardcopy-Routine zum Ausdruck von Buffer ------------------------------------------------------*)

Procedure DruckeFenster(TabulatorStr :String255; (* Steuercode, der die Verschiebung am Rand bewirkt.    *)
                        Invers,                  (* Bei true wird das Bild invertiert.                   *)
                        Farbe       :Boolean;    (* Flag, ob 'Farb'-Druck                                *)
                        SchrittX, SchrittY,      (* Richtung des Bildes (±1); bewirkt Spiegelung         *)
                        VergrX, VergrXSchritt,   (* Vergrößerung / Punktdichte in X-Richtung             *)
                        VergrY, VergrYSchritt,   (*      "       /      "      in Y-Richtung                *)
                        Nadeln                   :Integer);

  Var InvMaske,
      a,i,j,k,dx,dy,bx, lfmul,
      GZeile, AnfangY, EndeY, vy,
      DruZeile, DruZMax,
      X, AnfangX, EndeX, vx :Integer;
      DBuf                  :Array[0..2,0..1440] of Byte; (* Hier wird der Druck-Code aufgebaut          *)
      doppelt8              :Boolean; (* Flag, ob das Bild mit doppeltem Anschlag gedruckt werden soll;  *)
                                                                          (*\nur bei 8-Nadeln. *)
  Const FarbTabelle :Array[1..3,0..9,0..8] of Byte =   (* Bitmuster für die Schraffuren                  *)
   (((0,0,0,0,0,0,0,0,0),(0,1,0,0,1,0,0,1,0),(0,0,0,0,0,0,0,0,0),(0,0,1,0,0,1,0,0,1),(0,0,0,0,0,0,0,0,0),
     (1,0,0,1,0,0,1,0,0),(0,0,0,0,0,0,0,0,0),(0,1,0,0,1,0,0,1,0),(0,0,0,0,0,0,0,0,0),(0,0,1,0,0,1,0,0,1) ),
    ((1,0,0,1,0,0,1,0,0),(0,1,0,0,1,0,0,1,0),(0,0,1,0,0,1,0,0,1),(1,0,0,1,0,0,1,0,0),(0,1,0,0,1,0,0,1,0),
     (0,0,1,0,0,1,0,0,1),(1,0,0,1,0,0,1,0,0),(0,1,0,0,1,0,0,1,0),(0,0,1,0,0,1,0,0,1),(1,0,0,1,0,0,1,0,0) ),
    ((1,1,1,1,1,1,1,1,1),(1,1,1,1,1,1,1,1,1),(1,1,1,1,1,1,1,1,1),(1,1,1,1,1,1,1,1,1),(1,1,1,1,1,1,1,1,1),
     (1,1,1,1,1,1,1,1,1),(1,1,1,1,1,1,1,1,1),(1,1,1,1,1,1,1,1,1),(1,1,1,1,1,1,1,1,1),(1,1,1,1,1,1,1,1,1) ));
       lfStr :String[2]   = #27'J';  (* Steuercode für Zeilenvorschub in 1/3-Punkt-Abstand                *)

  Begin  (* Prozedur DruckeFenster *)
  dx := Buffer[3] shl 8 + Buffer[2];  (* Größe des zu druckenden Bildes;                                  *)
  dy := Buffer[5] shl 8 + Buffer[4];  (* wird direkt aus Buffer abgelesen                                 *)
  TabulatorStr := TabulatorStr + chr(lo(dx*VergrX)) + chr(hi(dx*VergrX)); (*Anzahl Grafikpunkte/Druckzeile*)
  if Farbe then bx := (dx + 3) div 4  (* Anzahl Bytes in Buffer pro Grafikzeile                           *)
           else bx := (dx + 7) div 8;
  if Farbe then dx := dx * VergrX;    (* Bei Farbe andere Zählweise                                       *)
  if SchrittX = 1 then Begin AnfangX := 0; EndeX := dx; End   (* Anfangs- und Endwerte entsprechend der    *)
                  else Begin AnfangX := dx - 1; EndeX := -1; End;              (* Spiegelung einstellen. *)
  SchrittY := -SchrittY;              (* Auf dem Drucker wird von oben nach unten gedruckt.               *)
  if SchrittY = 1 then Begin AnfangY := 0; EndeY := dy; End   (* Anfangs- und Endwerte für die Y-Richtung *)
  End;

(*------ Hardcopy-Routine zum Ausdruck von Buffer ------------------------------------------------------*)
```

Bild 7. Die Turbo-Pascal-Prozedur zum Drucken, Speichern und Laden von Grafikbildern

```
                else Begin AnfangY := dy - 1; EndeY := -1; End;                    (* einstellen *)
if Nadeln = 24 then Begin DruZMax := (24 div VergrY) * VergrY; (* Anzahl Zeilen, die auf einmal gedruckt*)
                         doppelt8 := false;                                          (*\werden. *)
                         lfmul := 1; End (* Zeilenvorschub geschieht im kleinsten Raster          *)
              else Begin DruZMax := 8;  (* Für 8-Nadeln ist die Vergrößerung nicht voll implementiert  *)
                         lfmul := 3;    (* Im 8-Nadel-Modus immer 3 Schritte Zeilenvorschub          *)
                         if VergrYSchritt = 1 then doppelt8 := false  (* doppelter Anschlag: nein    *)
                            else Begin doppelt8 := true; VergrYSchritt := 1; End; (*  "    : ja     *)
                   End;
if Invers then InvMaske := $FF      (* Xor-Maske, über die die Bildinvertierung gesteuert wird.      *)
          else InvMaske := 0;

GZeile := AnfangY;
repeat                              (* Durchzählen der Bildschirmzeilen                              *)
   for i := 0 to dx-1 do Begin DBuf[0,i] := 0; DBuf[1,i] := 0; DBuf[2,i] := 0; End;   (* DBuf löschen  *)
   DruZeile := 0;                   (* Zähler für die Drucker-Grafikzeilenn                          *)
   repeat                           (* Aufbau einer Druckerzeile in DBuf                             *)
      if Farbe then                 (* Bei 'Farb'- und s/w-Druck verschieden                         *)
         Begin                      (* 'Farbe'  (S/W mit erhöhtem Kontrast)                          *)
         X := 0;                    (* Zähler der Bildschirmpunkte in X-Richtung                     *)
         repeat
            a := ((Buffer[6 + GZeile*bx + X shr 2] shr (2*(3 - (X and $3)))) xor InvMaske) and $3;
                                    (* In  a  steht nun das Bitmuster des nächsten Bildschirmpunktes.*)
            if a > 0 then           (* Bei Hintergrund würden sowieso nur Nullen eingeschoben        *)
               for vy := 0 to VergrY - 1 do    (* Durchzählen der Druckzeilen des Punktes            *)
                  Begin
                  k := (DruZeile+vy) div 8; j := 7 - ((DruZeile+vy) mod 8);   (* Position in DBuf    *)
                  for i := 0 to VergrX - 1 do    (* Einschieben der Punkte aus der Farbtabelle      *)
                     DBuf[k,X*VergrX+i] := DBuf[k,X*VergrX+i]  +  FarbTabelle[a,vy,i] shl j;
                  End;
            X := X + 1;             (* Nun nächster Bildschirmpunkt                                  *)
         until X = dx div VergrX;   (* bis eine Bildschirmzeile fertig in DBuf übertragen ist        *)
         DruZeile := DruZeile + VergrY; (* Druckerzeilen weiterzählen                                *)
         End
      else                          (* Schwarz/weiß                                                  *)
         Begin                      (* Hier werden die Zeilen für die Vergr. nacheinander durchlaufen.*)
         vy := 0;                   (* vy zählt die Druckergrafikzeilen pro Bildschirmzeile.         *)
         repeat
            X := 0;
            k := DruZeile div 8; j := 7 - (DruZeile mod 8);  (* Position in DBuf: k=Bytenr., j=Bitnr.*)
            repeat
               a := Buffer[6 + GZeile*bx + X shr 3] xor InvMaske;  (* Ein Byte aus dem Buffer holen  *)
               for i := 7 downto 0 do  (* und in DBuf übertragen. Die Vergrößerung in X-Richtung     *)
                  Begin                (* geschieht beim s/w-Druck während der Übertragung zum Drucker.*)
                  DBuf[k,X] := DBuf[k,X] + ((a shr i) and 1) shl j;
                  X := X + 1;
                  End;
            until X >= dx;
            DruZeile := DruZeile + VergrYSchritt;  (* Entsprechend der Punktdichte Zeilen auslassen.*)
            vy := vy + VergrYSchritt;
         until vy >= VergrY;
         DruZeile := DruZeile - (vy - VergrY);  (* DruZeile korrigieren, damit die Vergrößerung und  *)
         End;                       (* die Punktdichte nicht unbedingt aufgehen müssen.              *)
      GZeile := GZeile + SchrittY;  (* Nun ist eine Bildschirmzeile in DBuf übertragen worden.       *)
   until (GZeile = EndeY) or (DruZeile = DruZMax); (* Weitermachen,bis Ende erreicht oder DBuf voll ist.*)
                                    (* Nun DBuf ausdrucken                                           *)
   write(lst,#13,TabulatorStr);     (* Zuerst linken Rand + Grafikbefehl-Anfang übersenden.          *)
   X := AnfangX;                    (* Zähler der Bytes in DBuf                                      *)
   vx := 0;                         (* Zähler für die Vergrößerung                                   *)
   repeat
      if Farbe then  write(lst,chr(DBuf[0,X]), chr(DBuf[1,X]), chr(DBuf[2,X]))
      else repeat
           if Nadeln = 24
           then if (vx mod VergrXSchritt) = 0   (* Drucken der Punkte (=0) oder Zwischenraum (<>0)  *)
                then write(lst, chr(DBuf[0,X]), chr(DBuf[1,X]), chr(DBuf[2,X]))
                else write(lst, #0#0#0)  (* Hier wird der Punktabstand in X-Richtung berücksichtigt. *)
           else write(lst,chr(DBuf[0,X]));  (* 8-Nadel-Druck: Nur ein Byte                           *)
           vx := vx + 1;
           until (vx mod VergrX) = 0;
      X := X + SchrittX;
   until X = EndeX;
```

```
   if doppelt8 then Begin           (* Doppelter Druck im 8-Nadel-Modus                              *)
                    write(lst,lfStr,#1,#13,TabulatorStr); (* Bereits 1/3 Punkt Zeilenvorschub        *)
                    X := AnfangX; vx := 0;
                    repeat write(lst,chr(DBuf[0,X])); X := X + SchrittX; until X = EndeX;
                    write(lst,lfStr + chr(DruZeile*lfmul - 1));  (* Rest des Zeilenvorschubs          *)
                    End
               else write(lst, lfStr + chr(DruZeile*lfmul));  (* Normaler Zeilenvorschub             *)
   until GZeile = EndeY;            (* Wiederholen, bis das ganze Fenster gedruckt ist.              *)
   End;  (* Prozedur DruckeFenster *)

(*------ Eingabe und Aufbereitung der Daten für DruckeFenster ------------------------------------*)

Procedure Drucke;

   Var c,v,s       :Char;           (* zur Eingabekontrolle                                          *)
       XVer        :Real;           (* Eingabe der Verschiebung des Bildes vom linken Rand           *)
       MultiX, MultiY, PuAbstX, PuAbstY,(* Zur Eingabe und Entwicklung der Parameter für DruckeFenster*)
       tt, i, VergX, VergY, VSchrX, VSchrY, Nadeln :Integer;
       TabSt       :String255;
       Invers,                      (* Flag, ob das Bild invertiert werden soll.                     *)
       Farbe       :Boolean;        (* Flag ob 'Farbe' gedruckt werden soll                          *)

   Const DruckModus :Array[1..12,1..6] of Byte = (* Tabelle mit dem Druckmodus für die entspr. Vergrößerung *)
                 ((1,1,1,1,1,1), (1,2,1,1,1,1), (1,1,3,1,1,1), (1,2,1,4,1,1), (1,1,1,1,1,1), (1,2,3,2,1,6),
                  (1,1,1,1,1,1), (1,2,1,4,1,2), (1,1,3,1,1,3), (1,2,1,2,1,2), (1,1,1,1,1,1), (1,2,3,4,1,6) );
         DruckBefehle :Array[1..6] of String[4] = (#27'*'#40, #27'*'#39, #27'*'#33, #27'*'#38, '', #27'*'#32);
         Grafmode_8Nadeln :String[2] = #27'L'; (* Steuercode für 8-Nadel Druckmodus 1/120 Zoll-Raster  *)
         DruckEmpfehlung :Array[1..12] of String[5] =
                        ( '1','2','3','2,4','(2,3)','2,3,4','(2,3)','2,4','3','2,4','(2,3)','2,3,4');
         TabCode :String[2] = #27'\';   (* Steuercode um den Druckkopf horizontal zu verschieben: 1/180 Zoll*)

   Begin  (* Prozedur Drucke *)       (* Einstellen der Druckparameter                                 *)
   PutPic(Buf1,0,199);              (* Ganzes Bild wieder anzeigen                                   *)
   GetPic(Buffer, X1, Y1, X2, Y2);  (* und entsprechenden Ausschnitt in Buffer einlesen              *)
   if GrafMode = 2 then             (* Zuerst festlegen, ob Kontrast- oder Hires-Druck               *)
      Begin                         (* Grafmode=2: Bildschirm ist im Farbmodus (halbe Auflösung)     *)
      Farbe := true;
      write25('Farbmodus:  2=doppelt   3=Standard     4=vergrößert    5=''schwarz/weiß''  ?');
      repeat c := InKey; if Esc then exit; until c in ['2'..'5'];
      if c = '5' then Begin         (* Farbbild in Schwarz/Weiß-Bild umwandeln                       *)
                      Buffer[0] := 1;   (* Modus = 1                                                 *)
                      tt := (Buffer[3] shl 8  +  Buffer[2]) * 2; (* In X-Richtung ist die Punktmenge dann *)
                      Buffer[2] := lo(tt); Buffer[3] := hi(tt);  (* doppelt so groß.               *)
                      Farbe := false;   (* Im weiteren wie Schwarz/Weiß behandeln                    *)
                      End;
      End
   else Farbe := false;             (* Bei Grafmode=1 (=Hires) in jedem Fall keine Farbe             *)

   if Farbe then                    (* Nun Eingabe und Einstellung der Bildvergrößerung              *)
      Begin                         (* zunächst bei Farbdruck                                        *)
      case c of                     (* c wurde ja oben bereits eingegeben                            *)
        '2': Begin VergX := 2; VergY := 2; End; (* fest eingestellte Vergrößerung: doppelt            *)
        '3': Begin VergX := 3; VergY := 3; End; (*                                  Standard          *)
        '4': Begin                  (* variable Vergrößerung                                         *)
             write25('Vergrößerung in X-Richtung (2-9) '); repeat readXY(33,25,VergX) until VergX in [2..9];
             write25('Vergrößerung in Y-Richtung (2-10) '); repeat readXY(34,25,VergY) until VergY in [2..10];
             End;
      End;
      Nadeln := 24;                 (* In jedem Fall 24-Nadel-Druckmodus                             *)
      MultiX := 2; PuAbstX := 2;    (* Dummy-Werte, damit der Druckmodus unten richtig eingestellt wird.*)
      End
   else
      Begin                         (* und bei Schwarz/Weiß-Druck                                    *)
      Nadeln := 24;                 (* Defaulteinstellung                                            *)
      write25('24-Nadeln:  1=komprimiert  2=doppelt  3=Standard  4=vergrößert  5=8-Nadeln   ?');
      repeat c := InKey; if Esc then exit; until c in ['1'..'5'];
      case c of
        '1','2','3': Begin tt := ord(c) - ord('0'); (* voreingestelte Werte: Vergrößerung 1,2 oder 3   *)
                           MultiX := tt; PuAbstX := tt; MultiY := tt; PuAbstY := 1; End;
        '4' : Begin
```

```
249             repeat write25('Vergrößerung X-Richtung (1..12)'); readXY(33,25,MultiX) until MultiX in [1..12];
250             repeat write25('Punktabstand X-Richt. ['+DruckEmpfehlung[MultiX]+']'); readXY(32,25,PuAbstX)
251             until PuAbstX in [1..6];
252             repeat write25('Vergrößerung Y-Richtung (1..24)'); readXY(33,25,MultiY) until MultiY in [1..24];
253             repeat write25('Punktabstand Y-Richtung (1..VY)'); readXY(33,25,PuAbstY)
254             until PuAbstY in [1..MultiY];
255             End;
256       '5' : Begin                       (* 8-Nadel-Druck                                              *)
257             Nadeln := 8;
258             write25('1=einfach  2=doppelt     ?');
259             repeat v := InKey; if Esc then exit; until v in ['1','2']; (* Auswertung folgt unten          *)
260             End;
261       End; (* case c of *)
262     End;                                (* Ende der farb-spezifischen Eingaben                        *)
263
264   write25('XPosition des Bildes (in cm) :'); read(XVer);
265   if Nadeln=8 then if round(XVer/(2.54/120)) > 240 then XVer := round(240*2.54/120);
266   Invers := false;                      (* Vorbelegung                                                *)
267   repeat
268     write25('1=normal 2=gesp.(X-A) 3=gesp.(Y-A) 4=2+3  5=verdoppeln(Y-Richt.)  9=invers  ?');
269     repeat s := InKey; if Esc then exit; until s in ['1'..'5','9'];
270     if s = '9' then Invers := true;
271   until s <> '9';                       (* Wenn Invertierung eingegeben wurde, dann Zeile nochmal anzeigen *)
272                         (* Nun Einstellen der Steuercodes für den linken Rand und die Grafikinitialisierung *)
273   case c of                             (* c stimmt auch bei Farbe                                    *)
274     '1'..'4':Begin                      (* 24-Nadel-Druck                                             *)
275             if not Farbe then           (* Bei Farbe sind die Vergrößerungswerte bereits eingestellt,  *)
276               Begin                     (* die Schrittweiten sind bei Farbe irrelevant                *)
277               VergY := MultiY;  VSchrY := PuAbstY;
278               VergX := MultiX div DruckModus[MultiX,PuAbstX];
279               VSchrX := PuAbstX div DruckModus[MultiX,PuAbstX];
280               End;
281             tt := round(XVer /(2.54/180)); (* Anzahl Punkte,die zur Verschiebung des Randes notwendig sind*)
282             TabSt := TabCode + chr(Lo(tt)) + chr(Hi(tt)) + DruckBefehle [DruckModus [MultiX, PuAbstX]];
283             End;
284     '5' : Begin
285           VergX := 1;  VSchrX := 1;  VergY := 1;  VSchrY := ord(v)-ord('0');
286           tt := round(XVer /(2.54/120)); TabSt := Grafmode_8Nadeln + chr(lo(tt)) + chr(hi(tt));
287           for i := 1 to tt do TabSt := TabSt + #0; (* linken Rand als 'weiße' Grafik drucken           *)
288           TabSt := TabSt + Grafmode_8Nadeln;
289           End;
290     End;
291
292   PutPic(Buf1,0,199);                   (* Ganzes Bild wieder anzeigen                                *)
293   GotoXY(1,25); write('Bitte einen Moment warten.');
294   case s of
295     '1' : DruckeFenster(TabSt,Invers,Farbe, 1, 1, VergX, VSchrX, VergY, VSchrY, Nadeln);
296     '2' : DruckeFenster(TabSt,Invers,Farbe, 1,-1, VergX, VSchrX, VergY, VSchrY, Nadeln);
297     '3' : DruckeFenster(TabSt,Invers,Farbe,-1, 1, VergX, VSchrX, VergY, VSchrY, Nadeln);
298     '4' : DruckeFenster(TabSt,Invers,Farbe,-1,-1, VergX, VSchrX, VergY, VSchrY, Nadeln);
299     '5' : Begin
300           DruckeFenster(TabSt,Invers,Farbe, 1, 1, VergX, VSchrX, VergY, VSchrY, Nadeln);
301           DruckeFenster(TabSt,Invers,Farbe, 1,-1, VergX, VSchrX, VergY, VSchrY, Nadeln);
302           End;
303     End;
304   End;  (* Prozedur Drucke *)
305
306 (*------ Speichern des Bildfensters ------------------------------------------------------------------*)
307
308 Procedure speichere;
309
310   Var Dat      :File of Puffer;
311       DatName  :String[64];
312
313   Begin
314   repeat
315     write25('Speichern - Dateiname :'); read(DatName);
316     if Datname='' then exit;            (* Bei leerer Eingabe abbrechen                               *)
317     assign(Dat,Datname);
318 (*$I-*)
319     rewrite(Dat);                       (* Datei öffnen; dabei testen, ob Schreiben in die Datei möglich *)
320 (*$I+*)
321   until ioresult = 0;
322   PutPic(Buf1,0,199);                   (* zuerst das Originalbild in den Bildschirmspeicher bringen  *)
323   GetPic(Buffer, X1, Y1, X2, Y2);       (* nun das markierte Fenster holen                            *)
324   write(Dat,Buffer);                    (* und abspeichern                                            *)
325   close(Dat);
326   End;  (* Prozedur speichere *)
327
328 (*------ Laden des Bildfensters ----------------------------------------------------------------------*)
329
330 Procedure lade;
331
332   Var Dat      :File of Puffer;
333       DatName  :String[64];
334
335   Begin
336   repeat
337     write25('Laden - Dateiname :'); read(DatName);
338     if Datname='' then exit;            (* Bei leerer Eingabe abbrechen                               *)
339     assign(Dat,Datname);
340     (*$I-*)  reset(Dat);  (*$I+*)       (* Datei öffnen                                               *)
341   until ioresult=0;                     (* So lange probieren, bis die Datei gefunden werden kann.    *)
342   read(Dat,Buffer);                     (* Datei lesen                                                *)
343   close(Dat);
344   X1 := 0; Y1 := 0;                     (* Altes Bild wird überschrieben                              *)
345   X2 := Buffer[3] shl 8 + Buffer[2] - 1;  (* Neue Fenster-Daten ermitteln                             *)
346   Y2 := Buffer[5] shl 8 + Buffer[4] - 1;
347   GrafMode := Buffer[0];
348   case GrafMode of                      (* Grafikmodus neu einstellen                                 *)
349     1: Hires;   2: GraphColorMode;  End;
350   PutPic(Buffer,0,Y2);                  (* Nun neues Bild anzeigen                                    *)
351   case GrafMode of                      (* Das ursprüngliche Bild wird überschrieben.                 *)
352     1: GetPic(Buf1, 0,0, 639,199);
353     2: GetPic(Buf1, 0,0, 319,199);
354     End;
355   Rechteck(X1,Y1, X2,Y2);               (* Neues Bild nun einrahmen                                   *)
356   End;  (* Prozedur lade *)
357
358 (*------ Hauptprogramm der Prozedur Hardcopy ---------------------------------------------------------*)
359
360   Begin  (* Prozedur Hardcopy *)
361   GetPic(Buf1,0,0,319,199);             (* Bestehendes Bild retten                                    *)
362   if Buf1[0] = 1 then Begin             (* Bildschirmmodus bestimmen --  1 = Hires                    *)
363                       GetPic(Buf1,0,0,639,199);
364                       GrafMode := 1;
365                       End
366                  else GrafMode := 2;    (* Farbmodus / Halbe Auflösung                                *)
367   repeat
368     PutPic(Buf1,0,199);                 (* Bild in jedem Fall neu anzeigen, um evt. Fehlermeld. zu entfernen*)
369     write25('<H>ardcopy  <S>peichern  <L>aden  <F>enster  <M>enue  ?');
370     Rechteck(X1,Y1,X2,Y2);              (* zu druckendes Fenster einrahmen                            *)
371     repeat ch := upcase(InKey); until (ch in ['H','S','L','F','M']) or Esc;
372     case ch of
373       'H': Begin drucke; Esc := false; End;
374       'S': speichere;
375       'L': lade;
376       'F': Begin                        (* zu druckendes Fenster verändern                            *)
377            write25('X1 =      Y1 =      X2 =      Y2 =    ');
378            GotoXY(6,25); write(X1); GotoXY(16,25); write(Y1);
379            GotoXY(26,25); write(X2); GotoXY(36,25); write(Y2);
380            ReadXY(5,25, X1);   ReadXY(15,25, Y1);   ReadXY(25,25, X2);  ReadXY(35,25, Y2);
381            End;
382       End;
383   until (ch = 'M') or Esc;
384   PutPic(Buf1,0,199);                   (* Bild wieder herstellen                                     *)
385   End;  (* Prozedur Hardcopy *)
386
387 (*                                Ende des Moduls Hardcopy                                            *)
388 (*****************************************************************************************************)
```

Peter Ibach

Flächenfüll-Algorithmus

Wir stellen hier ein Turbo-Pascal-Programm vor, mit dem beliebig große und beliebig geformte Figuren mit einer Farbe gefüllt werden. Das Programm wurde für die Hercules- und die IBM-Color-Karte angepaßt.

Mit der Procedure Fill (x,y:Integer);, die wir hier vorstellen (*Bild 1*), wird die gesamte beliebig große Fläche, in der (x,y) liegt, gefüllt. (x,y) muß vollständig von einem geschlossenen Streckenzug umgeben sein. Aufgrund der Geschwindigkeit wird keine weitere Abfrage gemacht. Liegt also keine geschlossene Fläche vor, kann die Farbe „auslaufen", was einen Systemabsturz zur Folge hat (Endlosschleife).

Der Algorithmus ist in Hinsicht auf Umfang und Geschwindigkeit optimiert worden. Damit er jedoch an die verschiedenen Hardwarevoraussetzungen anpaßbar bleibt, werden nur die Routinen Punktsetzen und Punktabfragen vorausgesetzt. Ein byteweises Füllen wäre natürlich vielfach schneller. Meine Absicht ist es, hiermit die Funktionsweise des Flächenfüllens darzulegen, wobei die individuelle Anpassung an die Hardware und z. B. Übertragung in Maschinencode dem einzelnen überlassen bleibt.

Ausgehend von (x,y) wird jeweils immer eine Linie bis zum linken und rechten Rand gefüllt. Die beiden äußersten Punkte werden gemerkt. Bei der nächsten Linie wird nach links und rechts bis zum Rand gezeichnet. Ist jedoch das Ende der vorhergehenden Zeile noch nicht überschritten, wird noch bis zu diesem Ende weiterhin überprüft, ob hier ein Flächenabschnitt ist. Wendet man dieses Verfahren nach oben und nach unten an, ist die Fläche mit Sicherheit vollständig ausgefüllt. Für jede Zeile wird eine Rekursion ausgeführt. Zur Minimierung der abzufragenden Punkte wird von jeder Zeile aus nur max. einmal nach oben, bzw. unten gegangen, nicht etwa für jeden Punkt der Linie. In die Rückrichtung, aus der man gekommen ist, wird nur dann zurückuntersucht, wenn die neue Zeile die alte Zeile an den Außenstellen überlappt. Zur Optimierung merkt sich das Programm, welche Linien bereits gezeichnet wurden und überprüft so keinen Punkt überflüssigerweise. Eine weitere wesentliche Geschwindigkeitssteigerung brächte das byteweise füllen. Ist ein Byte 0, wird es sofort auf 255 (11111111) gesetzt, ohne erst eine aufwendige Einzelpunktabfrage zu machen (ca. 2500 Punkte/s).

Zwei Beispiele

Falls die obige Erklärung zu kompliziert erscheint, zeigt *Bild 2* ein einfacheres Fill-Programm, das die Problematik veranschaulicht. Dieses Fill-Programm ist bereits vollständig funktionsfähig. Es hat jedoch noch zwei entscheidende Nachteile:

1. Es ist zu langsam (ca. 1000 Punkte/s).
2. Es benötigt für jeden Punkt 6 Bytes (x + y + Rückkehradresse = 6). Bei einer

```
(*Hercules Graphic Card Include File Driver                              *)
(*$I herc    <-Graphikerweiterung: Point  (x,y)          Punkt setzen    *)
(*                                 Dot    (x,y)          Punkt testen    *)
(*                                 Draw   (x1,y1,x2,y2,Farbe)  Linie     *)
(*                                 Drawto (x,y)             " fortsetzen *)
(*                                 Hires                  Graphikmodus.   *)
(*                                 Textmode               Textmodus       *)

(*************************************************************************)

procedure fill2 (xl,xr,y,dir:integer);   (*xl,xr=Eckpunkte der letzten Zeile*)
var xln,xrn:integer;                     (*y=Zeile, Dir=Suchrichtung*)
                                         (*xln,xrn=xl,xr neu*)
begin
   while dot(xl,y) do begin              (*Luecken suchen *)
      xl:=xl+1;
      if xl>xr then exit;
   end;
   xln:=xl; xrn:=xl+1;
   while not dot(xrn,y) do begin         (*Bits rechts fuellen*)
      point(xrn,y); xrn:=xrn+1;
   end;
   while not dot(xln,y) do begin         (*Bits links*)
      point(xln,y); xln:=xln-1;
   end;
   if xrn<xr then fill2(xrn,xr,y,dir)              (*Rekursion*)
    else fill2(xr,xrn-1,y-dir,-dir);
   if xln<xl then fill2(xln+1,xl,y-dir,-dir);      (*Ueberlappungskriterien*)
   fill2(xln+1,xrn-1,y+dir,dir);
end;

procedure fill (x,y:integer); (*startet fill2*)
begin fill2(x,x,y,1); end;

procedure drawshape1; (*Zeichnet eine moeglichst komplizierte Flaeche*)
begin
   draw(10,10,100,10,1);
   drawto(100,50);
   drawto(200,20);
   drawto(150,70);
   drawto(100,60);
   draw(100,60,100,100,1);
   drawto(10,100);
   drawto(10,10);
   draw(40,40,60,60,1);
   drawto(70,20);
   drawto(40,40);
   draw(10,80,50,90,1);
   drawto(98,98);
   drawto(80,20);
   drawto(10,80);
end;
```

Bild 1. Der Flächenfüll-Algorithmus wird hier in drei Beispielen angewandt

```
procedure drawshape2; (*Zeichnet eine andere Flaeche*)
begin
   draw(1,1,200,1,1);
   drawto(200,140);
   drawto(1,140);
   drawto(1,1);
end;

procedure fill3 (x,y:integer);   (*Primitives Fill*)
begin                            (*Achtung ! Hohe Recursionstiefe*)
   if not dot(x,y) then begin
      point(x,y);
      fill3(x+1,y);
      fill3(x,y+1);
      fill3(x-1,y);
      fill3(x,y-1);
   end;
end;

begin
   hires;               (*Graphik einschalten *)
   drawshape1;          (*Flaeche1 malen      *)
   drawshape2;          (*Flaeche2 malen      *)
   fill(30,30);         (*   1    ausfuellen  *)
   fill(130,100);       (*   2    ausfuellen  *)
   readln;              (*Warten              *)
   textmode;            (*Graphik aus         *)
end.
```

```
Procedure Fill ( x,y:Integer);
Begin
   If Punkt_nicht_gesetzt (x,y) then Begin      (* Solange bis Rand *)
      Setze_Punkt (x,y);                        (* Fuelle           *)
      Fill (x+1,y);  (* Rechts *)              (* in allen Richtungen*)
      Fill (x-1,y);  (* Links  *)
      Fill (x,y+1);  (* Unten  *)
      Fill (x,y-1);  (* Oben   *)
   End (* If *);
End (* Fill *);
```

Bild 2. Die einfachste und speicherintensivste Variante des Flächenfüll-Algorithmus

```
Procedure Fill (x,y:integer);
Var xl,xr:Integer;      (* Linke und Rechte Grenze der Zeile y*)
Begin
   xl:=x; xr:=x+1;      (* Fange bei x an *)
   While Punkt_nicht_gesetzt (xr,y) do begin
       setze_Punkt (xr,y);
       xr:=xr+1;
   end;                 (* Rechts zum Rand *)
   While Punkt_nicht_gesetzt (xl,y) do begin
       setze_Punkt (xl,y);
       xl:=xl-1;
   end;                 (* Links  zum Rand  *)
   For x:=xl to xr do Begin                    (* Prüfe für jeden Punkt*)
      If Punkt_nicht_gesetzt (x,y-1) then Fill (x,y-1);   (* Oben *)
      If Punkt_nicht_gesetzt (x,y+1) then Fill (x,y+1);   (* Unten *)
   End (* For x *);
End (* Fill *);
```

Bild 3. Schon etwas schneller und speichersparender geht es mit diesem Programm

```
type str=string[80];

var xcor,ycor,color:integer;
    mptr:record case byte of 0:(p:^byte); 1:(offs,segm:integer); end;
    ladr:array[0..347] of integer;

procedure clear;
begin fillchar(mem[mptr.segm:0],32767,0); end;

PROCEDURE mode(onof:byte);  (*1=text, 0=grafik ,Hardwareabhngig*)
const   table : ARRAY[0..1,0..12] OF Byte
              = ((130,$35,$2d,$2e,$07,$5b,$02,$57,$57,$02,$03,$00,$00),
                 (32,$61,$50,$52,$0f,$19,$06,$19,$19,$02,$0d,$06,$0a));
  VAR   I       : Byte;
  BEGIN
    onof:=onof mod 2; Port[$03bf]  := 3; port[$3b8]:=table[onof,0];
    FOR i:=0 TO 11 DO BEGIN
        Port[$3b4]:=i; Port[$3b5 ]:=table[onof,i+1];
    END;
END;

procedure hires;
var y:integer;
begin
   mode(0); for y:=0 to 347 do ladr[y]:=(y and 3) shl 13+y shr 2*90;
   mptr.segm:=$b800; clear; port[$3b8]:=138;
end;

procedure textmode;
begin mode(1); delay(300); port[$03b8]:=40; end;

procedure point(x,y:integer);          (*Setzt x,y nach Color*)
begin
   mptr.offs:=ladr[y]+x shr 3;
   case color of
      0:mptr.p^:=mptr.p^ and not(128 shr (x and 7));
      1:mptr.p^:=mptr.p^ or 128 shr (x and 7);
      2: mptr.p^:=mptr.p^ xor 128 shr (x and 7);
   end;
end;

procedure plot(x,y,c:integer);
begin color:=c; point(x,y); xcor:=x; ycor:=y; end;

function dot(x,y:integer):boolean; (*true=set*)
begin
   mptr.offs:=ladr[y]+x shr 3; dot:=odd(mptr.p^ shr (not x and 7));
end;

procedure drawto(x,y:integer);         (*Linie von xcor,ycor nach x,y*)
var i,dx,dy,xstep,ystep,e:integer;     (*nach Bresenham Algorithmus  *)
begin
      dy:=abs(y-ycor); dx:=abs(x-xcor); e:=dx-dy;
      if x>xcor then xstep:=1 else xstep:=-1;
      if y>ycor then ystep:=1 else ystep:=-1;
      while (xcor<>x) or (ycor<>y) do begin
         if e>=0 then begin xcor:=xcor+xstep; e:=e-dy; end;
         if e<0 then begin ycor:=ycor+ystep; e:=e+dx; end;
         point(xcor,ycor);
      end;
end;

procedure draw(x1,y1,x2,y2,c:integer);  (*Linie von x1,y1 nach x2,y2 Farbe c*)
begin plot(x1,y1,c); drawto(x2,y2); end;
```

Bild 4. Grafik-Treiber für die Hercules-Karte

```
(* Anpassung an Graph.p Treiber *)

procedure point(x,y:integer);
begin plot(x,y,1); end;

function dot (x,y:integer):boolean;
begin if getdotcolor(x,y)=1 then dot:=true else dot:=false; end;

procedure drawto(x,y:integer);
begin draw(xcor,ycor,x,y,1); end;
```

Bild 5. Anpassung an die IBM-Color-Karte

kleinen Fläche von 100 × 100 Punkten benötigt es 600 000 Bytes Stack!

Entwickelt man das Programm geringfügig weiter, erhält man das in *Bild 3* dargestellte Programm.

Dieses Programm benötigt nur noch 10 Bytes pro Zeile, da immer erst eine ganze Zeile gefüllt wird, bevor die Rekursion erfolgt. Das Prinzip ist ganz einfach und stellt den Grundstock für den optimierten Fill-Algorithmus dar. Es wird eine Zeile nach links und rechts gefüllt, jeweils bis der Rand erreicht ist. Nun prüft man vom äußersten Punkt links bis zum äußersten Punkt rechts, ob oberhalb oder unterhalb noch ein zu füllender Punkt liegt.

Ist dies der Fall, muß hier noch gefüllt werden. Leider wird auch hier jeder Punkt sehr häufig abgefragt, da die nächste Zeile wieder die bereits gezeichneten Punkte überprüft.
Die in Bild 1 erforderliche Grafikerweiterung für die Hercules-Grafik-Karte ist in *Bild 4* aufgelistet. Die Anpassung an den Grafiktreiber für die IBM-Color-Karte ist in *Bild 5* zu sehen.

des Zehnerblocks die entsprechenden Zeichenfolgen ausgegeben werden, und der Tabelle mit den Zeichenfolgen. Zunächst wird AH mit dem Funktionscode 09H (Drucke Zeichenkette ab DX bis $) geladen, das DX-Register mit 200H (Anfang der Tabelle) und das CX-Register mit 0BH (Schleifenzähler für die 11 Tasten). Darauf erfolgt der DOS-Aufruf 21H. Da jede Zeichenkette 13 Zeichen lang ist, wird DX um 0DH bei jedem Schleifendurchlauf erhöht. Beendet wird das Programm mit INT 20H.
In *Bild 2* ist das Programm und sein Speicherauszug wiedergegeben. *Bild 3* zeigt die Tabelle der Grafikzeichen (ab 200H) sowie die Tabelle der fettgedruckten Grafikzeichen (ab 300H). Falls Sie diese bevorzugen oder diese aus Kompatibilitätsgründen mit anderen Programmen wählen, müssen Sie das DX-Register mit 0300H laden. Interessant ist an diesem Konzept, daß die Tabelle, um noch weitere Zusatz- bzw. Umbelegungen einzuschließen, beliebig nach unten erweitert werden kann. Sie müssen nur den Zähler auf den entsprechenden Wert setzen. Damit das Programm jederzeit zur Verfügung steht, sollte das Programm bereits von AUTOEXEC.BAT aufgerufen werden. *Jürgen H. Schmid*

Grafiksymbole beim PC-D

Die PC-Grafiksymbole eignen sich gut zum Gestalten von Texten und Tabellen aller Art. Sie sind jedoch bei der Standard-Tastaturbelegung nur umständlich erreichbar. Abhilfe schafft das kurze Maschinenprogramm GRAFIX.COM, das in leicht einprägsamer Weise den Alt-Shift-Zehnerblock mit Sonderzeichen belegt (*Bild 1*).

```
    7  8  9
    ┌  ┬  ┐
4   ├  ┼  ┤  5
    └  ┴  ┘
    1  2  3

0   |  "-"  _
```

Bild 1. Eine sinnvolle Belegung des Alt-Shift-Zehnerblocks

Diese Umbelegung ist beim Siemens PC-D leicht möglich. Pro Taste wird eine Zeichenfolge ausgegeben, die wie folgt aufgebaut ist:
'Esc ['
Dezimalzahl der Taste im ASCII-Code (hier erweiterter Code, mit '0' beginnend)
Dezimalzahl des neuen Zeichens im ASCII-Code
'p': Zeichen für benutzerdefinierte Taste
'$': schließt die Zeichenkette ab
Im Wesentlichen besteht das Programm aus einer Schleife, in der bei den Tasten

```
u 100 110
0FAA:0100 B409          MOV     AH,09
0FAA:0102 BA0002        MOV     DX,0200
0FAA:0105 B90B00        MOV     CX,000B
0FAA:0108 CD21          INT     21
0FAA:010A 83C20D        ADD     DX,+0D
0FAA:010D E2F9          LOOP    0108
0FAA:010F CD20          INT     20


d 100 110
0FAA:0100  B4 09 BA 00 02 B9 0B 00-CD 21 83 C2 0D E2 F9 CD   4.:..9..M!.B.byM
0FAA:0110  20
```

Bild 2. So einfach können beim Siemens PC-D Tasten umbelegt werden

```
d 200 28e
0FAA:0200  1B 5B 30 3B 31 32 39 3B-31 37 39 70 24 1B 5B 30   .[0;129;179p$.[0
0FAA:0210  3B 31 32 30 3B 31 39 32-70 24 1B 5B 30 3B 31 32   ;120;192p$.[0;12
0FAA:0220  31 3B 31 39 33 70 24 1B-5B 30 3B 31 32 32 3B 32   1;193p$.[0;122;2
0FAA:0230  31 37 70 24 1B 5B 30 3B-31 32 33 3B 31 39 35 70   17p$.[0;123;195p
0FAA:0240  24 1B 5B 30 3B 31 32 34-3B 31 39 37 70 24 1B 5B   $.[0;124;197p$.[
0FAA:0250  30 3B 31 32 35 3B 31 38-30 70 24 1B 5B 30 3B 31   0;125;180p$.[0;1
0FAA:0260  32 36 3B 31 36 39 70 24-1B 5B 30 3B 31 32 37 3B   26;169p$.[0;127;
0FAA:0270  31 39 34 70 24 1B 5B 30-3B 31 32 38 3B 31 37 30   194p$.[0;128;170
0FAA:0280  70 24 1B 5B 30 3B 31 37-36 3B 31 39 36 70 24      p$.[0;176;196p$


d 300 38e
0FAA:0300  1B 5B 30 3B 31 32 39 3B-31 38 36 70 24 1B 5B 30   .[0;129;186p$.[0
0FAA:0310  3B 31 32 30 3B 32 30 30-70 24 1B 5B 30 3B 31 32   ;120;200p$.[0;12
0FAA:0320  31 3B 32 30 32 70 24 1B-5B 30 3B 31 32 32 3B 31   1;202p$.[0;122;1
0FAA:0330  38 3B 70 24 1B 5B 30 3B-31 32 33 3B 32 30 34 70   88p$.[0;123;204p
0FAA:0340  24 1B 5B 30 3B 31 32 34-3B 32 30 36 70 24 1B 5B   $.[0;124;206p$.[
0FAA:0350  30 3B 31 32 35 3B 31 38-35 70 24 1B 5B 30 3B 31   0;125;185p$.[0;1
0FAA:0360  32 36 3B 32 30 31 70 24-1B 5B 30 3B 31 32 37 3B   26;201p$.[0;127;
0FAA:0370  32 30 33 70 24 1B 5B 30-3B 31 32 38 3B 31 38 37   203p$.[0;128;187
0FAA:0380  70 24 1B 5B 30 3B 31 37-36 3B 32 30 35 70 24      p$.[0;176;205p$
_
```

Bild 3. Speicherauszug der beiden Grafikzeichen-Tabellen

Dieter Strauß

Ein Blick in das Environment

Die DOS-Programmumgebung sinnvoll eingesetzt

Mit den Informationen, die in der DOS-Programmumgebung – oft auch Environment genannt – stehen, kann das standardmäßige Verhalten eines Programms geändert werden. Bevor wir in die Tiefe des Betriebssystems hinabsteigen, klären wir, wie bereits Standard-DOS-Befehle die Programmumgebung ändern.

Viele DOS-Anwender können mit dem Begriff Environment wenig anfangen. Spätestens wenn im eigenen System eine Festplatte ihren Dienst versieht, sollte man sich mit dem Environment befassen. Mit zunehmender Zahl an Unterverzeichnissen verliert man leicht den Überblick über die Dateien, die auf der Festplatte abgespeichert sind, und die Frage: „In welchem Unterverzeichnis habe ich diese Datei nur abgelegt?" taucht immer häufiger auf. Zum Glück kennt MS-DOS den Befehl Path, mit dem ein direkter Zugriffspfad auf externe DOS-Befehle, EXE-, COM- und BAT-Dateien eingerichtet werden kann. Um den Zugriffspfad nicht immer nach dem Booten mühsam von Hand eingeben zu müssen, sollte man diesen Befehl in die AUTOEXEC.BAT-Datei schreiben. Damit ist bereits der erste Schritt zur bewußten Anwendung des Environments getan.

In *Bild 1* ist ein kurzes Beispiel mit einer AUTOEXEC.BAT-Datei aufgeführt. Der in der AUTOEXEC.BAT-Datei verwendete Befehl APPEND steht erst ab DOS 3.2 zur Verfügung. Mit dem APPEND-Befehl wird ein Suchpfad für Nichtprogrammdateien definiert.

Die Informationen die DOS, mit den Befehlen PATH, PROMPT und COMSPEC (steht in der CONFIG.SYS-Datei) erhält, werden im sogenannten Master Environment Block abgespeichert. Zusätzlich können mit dem SET-Befehl weitere Eintragungen vorgenommen werden, so lange diese der Syntax SET [variable=[string]] entsprechen. Ein Eintrag im Environment übergibt einem Programm zusätzliche Informationen. Im Allgemeinen sind dies Informationen, die von mehr als einem einzigen Programm benötigt werden. COMSPEC gibt z. B. an,

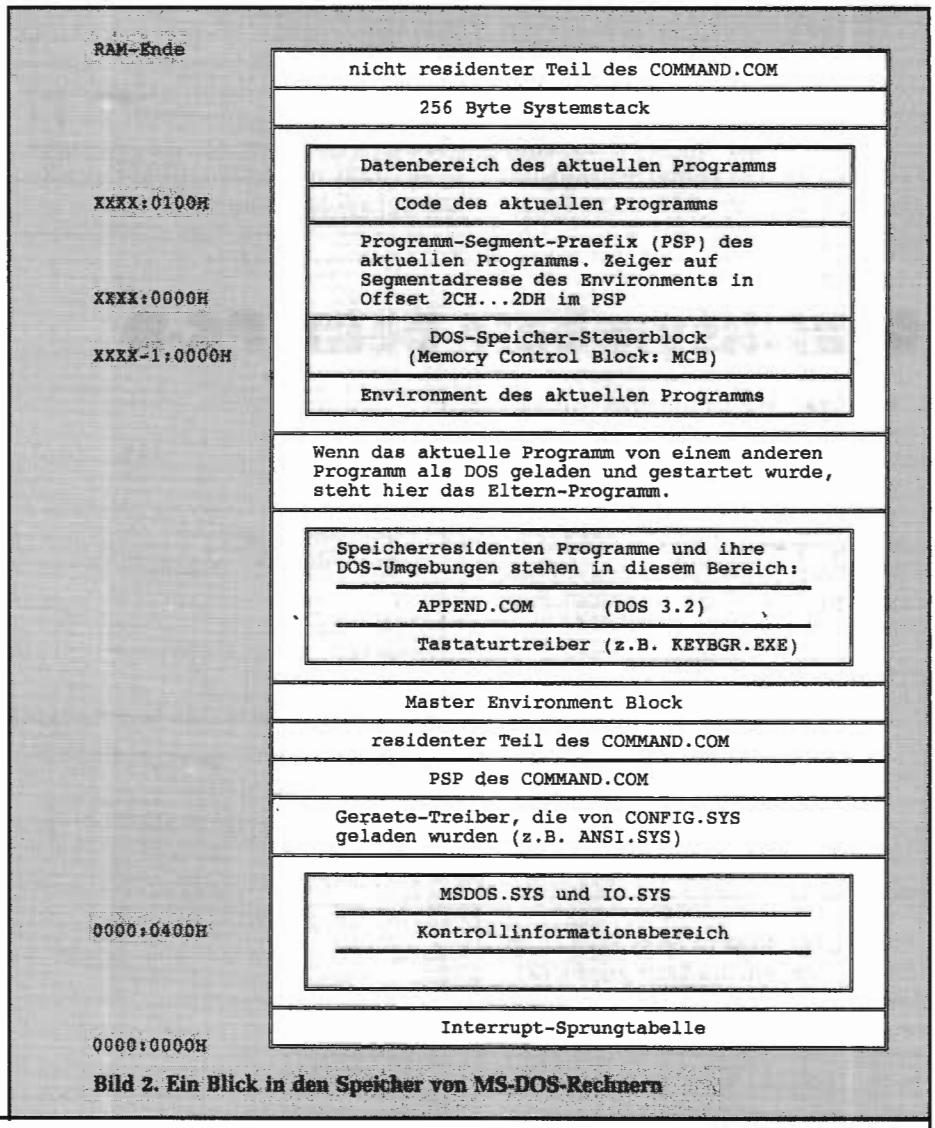


Bild 2. Ein Blick in den Speicher von MS-DOS-Rechnern

```
keybgr                                  Deutschen Tastaturtreiber installieren
path c:\;c:\euro;c:\strauss\programm;    Suchpfad für Programme und Stapeldateien
append c:\strauss\text;c:\strauss\unix;  Suchpfad für Textdateien
prompt=$p$g                             Der Prompt zeigt das aktuelle Directory mit an
set mc=bleibt mc                        Ein Werbeeintrag soll ins Environment
```

Bild 1. Beim Booten werden die Befehle der AUTOEXEC.BAT-Datei ausgeführt

wo der Kommandoprozessor (COMMAND.COM) liegt.

Der Master Environment Block sitzt oberhalb des residenten Teils des DOS-Kommandoprozessors (*Bild 2*). Mit dem SET-Befehl kann die DOS-Umgebung abgefragt und interaktiv geändert werden.

Wenn einem bereits bestehenden Suchpfad ein neuer hinzugefügt werden soll, ist es lästig, mit dem SET-Befehl wieder den gesamten Suchpfad einzutippen. Da Stapeldateien direkt auf den Master Environment Block zugreifen und diesen auch verändern können, hilft uns eine kleine Stapeldatei weiter, die SETPATH.BAT genannt sei. Der Inhalt dieser Stapeldatei lautet:
PATH=%PATH%;%1

Zum Hinzufügen eines Suchpfades können wir z.B. SETPATH C:\rohde\programm\mc1_87 eingeben.

Wie ist das Environment aufgebaut?

Die einzelnen Einträge im Environment stehen als ASCIIZ-String im Speicher, d. h., ein String ist mit einem Null-Byte abgeschlossen. Das Ende des Environments ist mit zwei Null-Byte markiert. Hinter diesem Eintrag befindet sich ab DOS 3.0 eine Kopie der Dateibezeichnung mit Laufwerk und Zugriffspfad. Ein Programm kann damit leicht feststellen, von welchem Programm es aufgerufen wurde. In *Bild 3* ist dargestellt, wie das Environment im Speicher aussieht. Der letzt gültige Eintrag ist DRUCKER-.COM. Der Rest ist „Datenmüll" aus vorherigen Programmen.

Manipulieren des Environments

Wenn ein Programm vom Kommandoprozessor aufgerufen wird, erhält dieses eine Kopie des Master Environment Blocks. Ein Programm, das von einem anderen Anwenderprogramm gestartet wird, erhält normalerweise eine Kopie der DOS-Umgebung des aufrufenden Programms (parent process). Das Laden und Ausführen eines Programms übernimmt die Exec-Funktion (DOS-Funktion 4BH, Unterfunktion 00H). In *Bild 4* ist die Exec-Funktion kurz illustriert. Soll das aufgerufene Programm (child process) eine andere Umgebung als das aufrufende Programm erhalten, so muß im Parameterblock der Funktion 4BH an der Offsetadresse 00H...01H ein Zeiger auf den Speicherbereich mit der gewünschten DOS-Umgebung vorhanden sein. Falls lediglich die DOS-Umgebung des aufrufenden Programms übernommen werden soll, muß an dieser Stelle 0000H stehen. Alle Änderungen, die von einem Programm an seiner DOS-Umgebung durchgeführt wurden, gehen jedoch nach dem Programmende verloren!

Wie findet ein Programm sein Environment?

Unter MS-DOS gehört zu jedem Programm ein Vorspann – der Programm-Segment-Präfix (PSP). Darin stehen Einträge, wie z.B. die Argumente der Befehlszeile und die Adresse, welche den nächsten Befehl nach dem Programmende beinhaltet. Unter den Offset-Adressen 2CH...2DH ist ein Zeiger auf die Segment-Adresse der DOS-Umgebung des jeweiligen Programms abgelegt. Unser Beispiel in Bild 3 zeigt einen Teil des PSP, der zum speicherresidenten Programm DRUCKER.COM [1] gehört. Der Zeiger auf die DOS-Umgebung ist zwei

```
DEBUG
-d 1000:7D60 7F2F
1000:7D60  82 08 68 17 68 17 0B 06-68 17 46 00 45 04 B8 00   ..h.h...h.F.E.8.

              Zeiger auf den zugehörigen PSP: 17E0:0000H bzw. 1000:7E00H

1000:7D70  4D E0 17 07 00 21 BE A2-07 E8 5F 00 EB D4 33 C9   M`...!>".h_.kT3I   Speicherkontrollblock
              └──┘ └──Speicherbereichslänge in Paragraphen
1000:7D80  43 4F 4D 53 50 45 43 3D-43 3A 5C 43 4F 4D 4D 41   COMSPEC=C:\COMMA   Environment
1000:7D90  4E 44 2E 43 4F 4D 00 50-41 54 48 3D 43 3A 5C 45   ND.COM.PATH=C:\E   zu DRUCKER.COM
1000:7DA0  55 52 4F 3B 43 3A 5C 53-54 52 41 55 53 53 5C 50   URO;C:\STRAUSS\P
1000:7DB0  52 4F 47 52 41 4D 4D 3B-00 50 52 4F 4D 50 54 3D   ROGRAMM;.PROMPT=
1000:7DC0  24 70 24 67 00 00 01 00-63 3A 5C 73 74 72 61 75   $p$g....c:\strau
1000:7DD0  73 73 5C 70 72 6F 67 72-61 6D 6D 5C 6D 63 31 5F   ss\programm\mc1_
1000:7DE0  38 37 5C 44 52 55 43 4B-45 52 2E 43 4F 4D 00 C3   87\DRUCKER.COM.C
              Zeiger auf den zugehörigen PSP

1000:7DF0  4D E0 17 E0 17 72 65 63-74 20 44 4F 53 20 56 65   M`...rect DOS Ve   Speicherkontrollblock
              └──┘ └──Speicherbereichslänge in Paragraphen
1000:7E00  CD 20 66 18 00 9A F0 FE-1D F0 2F 01 1E 0D 3C 01   M f...p./...<.     PSP zu
1000:7E10  1E 0D EB 04 1E 0D 1E 0D-01 01 01 00 02 FF FF FF   ..k.............   DRUCKER.COM

           Zeiger auf Beginn des Environments:  17D8:0000H bzw. 1000:7D80H

1000:7E20  FF FF FF FF FF FF FF FF-FF FF FF FF D8 17 E6 FF   ............X.f.

1000:7E30  E0 17 14 00 18 00 E0 17-FF FF FF FF 00 00 00 00   `.....`.........
           -- 1 Zeile mit 00 wurde weggelassen --
1000:7E50  CD 21 CB 00 00 00 00 00-00 00 00 00 00 20 20 20   M!K..........
1000:7E60  20 20 20 20 20 20 20 20-00 00 00 00 00 20 20 20            .....
1000:7E70  20 20 20 20 20 20 20 20-00 00 00 00 00 00 00 00            ........
1000:7E80  00 0D 24 70 24 67 0D 72-61 75 73 73 5C 74 65 78   ..$p$g.rauss\tex
1000:7E90  74 3B 63 3A 5C 73 74 72-61 75 73 73 5C 75 6E 69   t;c:\strauss\uni
1000:7EA0  78 3B 0D 00 00 00 00 00-00 00 00 00 00 00 00 00   x;..............
           -- 5 Zeilen mit 00 wurden weggelassen --
1000:7F00  E9 56 07 54 FF 00 F0 FB-50 53 51 52 56 57 1E 06   iV.T..p{PSQRVW..   Programmbeginn
1000:7F10  B8 40 00 8E D8 EB 0E 90-07 1F 5F 5E 5A 59 5B 58   8@..Xk...._^ZY[X   von DUCKER.COM
1000:7F20  2E FF 2E 03 01 2E A2 6B-03 80 3E 00 01 01 75 03   ......"k..>...u.
```

Bild 3. Ein Speicherauszug mit Speicherkontrollblöcken, dem Environment und einem PSP

Byte lang. Da Intel-Prozessoren das niederwertige Byte auf der jeweilig niederen Adresse ablegen, steht im Speicherauszug das niederwertige Byte links vom höherwertigen. Das DEBUG-Programm verlangt jedoch die umgekehrte Reihenfolge. Um sich das Environment des Programms ansehen zu können, müssen wir in unserem Beispiel 17D8:0000 eingeben. Die Adresse 17D8:0000 ist übrigens identisch mit 1000:7D80! Zur Probe müssen Sie nur den Segmentteil der Adresse mit 10H multiplizieren (eine Null anhängen) und den Offset-Teil dazu addieren.

Diese Erkundungsreise in den Speicher kann leicht an anderen speicherresidenten Programmen nachvollzogen werden. Zum Auffinden eines speicherresidenten Programms eignet sich der Suchbefehl von DEBUG. Als Suchbegriff kann z.B. COMSPEC verwendet werden. Die Befehlssequenz und die möglichen Antworten des Systems lauten:

```
DEBUG
-s 1000:0000 ffff 'COMSPEC'
1000:1EC5
1000:7650
1000:7D80
```

Der letztgenannte Eintrag gehört in unserem Beispiel zum Environment des residenten Programms DRUCKER.COM. Da im voraus nicht ersichtlich ist, welcher der Einträge zum gesuchten Programm gehört, muß man sich mit dem Dump-Befehl die ausgegebenen Speicherstellen näher anschauen.

Die Einträge im Environment können zum Ändern der Laufzeit-Parameter eines Programms verwendet werden. Dazu muß das aufgerufene Progamm seine DOS-Umgebung nach bestimmten Einträgen durchsuchen. Wenn der Zeiger ab Offset-Adresse 2CH des Programm-Segment-Präfixes in das DS-Register übertragen wird, so kann das aufgerufene Programm mit dem Move-String-Befehl (MOVSB) des 8086-Assemblers seine DOS-Umgebung in einen Puffer kopieren. Bei diesem Befehl wird die „Quelle" (hier: der Environment-Block) mit dem Registerpaar DS:SI und das „Ziel" (hier: der Puffer) mit dem Registerpaar ES:DI spezifiziert. Mit dem Compare-String-Befehl (CMPSB) kann das Programm den Puffer nach Einträgen durchsuchen. Es darf nicht vergessen werden, die PSP-Adresse vorher zu sichern, da sonst das System abstürzt.

Vergrößern des Environments

Die Größe des Environments ist auf 160 Byte voreingestellt. Für viele Anwendungen ist dieser Platz zu knapp bemessen. Unter MS-DOS 3.2 kann mit dem folgenden Befehl in der CONFIG.SYS-Datei die Größe des Environments geändert werden:

```
SHELL=COMMAND.COM /P /E:x
```

Mit dem Schalter /E:x wird die Größe der DOS-Umgebung in Byte eingestellt. Ohne den Schalter /P lädt das System beim Neustart nicht die AUTOEXC.BAT-Datei. Bei älteren DOS-Version kann mit Microsofts SETENV.EXE-Programm die Größe des Environment geändert werden. Steht dieses Programm nicht zur Verfügung, so kann mit einem einfachen Patch des COMMAND.COM (im „MS-DOS Profi-Arbeitsbuch" [2] beschrieben) das Environment vergrößert werden.

Speichermanagement

Außer dem Environment und dem PSP sind in Bild 3 weitere interessante Details zu sehen: die Speicherkontrollblöcke. DOS führt die Speicherverwaltung mit der vorwärts verketteten Liste von Speicherkontrollblöcken durch. Ein Speicherkontrollblock fängt an einer Paragraph-Adresse (Adresse MOD 10H = 0) an und ist 16 Byte lang. Der Beginn des Blocks wird von DOS mit 4DH ('M') gekennzeichnet. In den nächsten zwei Byte steht die PSP-Segmentadresse des Prozesses, zu dem der Speicherbereich gehört, der nach dem Kontrollblock folgt. Der Inhalt der Offsetadressen 03H und 04H gibt die Länge des Speicherbereichs in Vielfachen von 16 Byte (Paragraph) an. Der Rest des Kontrollblocks ist für DOS reserviert. Das erste Byte des letzten Speicherkontrollblocks hat den Wert 5AH ('Z'). Aus Bild 3 ist klar ersichtlich, warum das Environment nicht während des Betriebs vergrößert werden kann: die Speicherkontrollblöcke würden überschrieben werden! Ihr System verabschiedet sich in diesem Fall mit der Meldung:
Memory allocation error
Cannot load command, system halted

Literatur

[1] *Arko B.*: Druckeranpassung unter MS-DOS. mc 1987, Heft 1, Seite 69...71
[2] *Smode D.*: Das große MS-DOS Profi-Arbeitsbuch. Franzis Verlag, München, 1987.

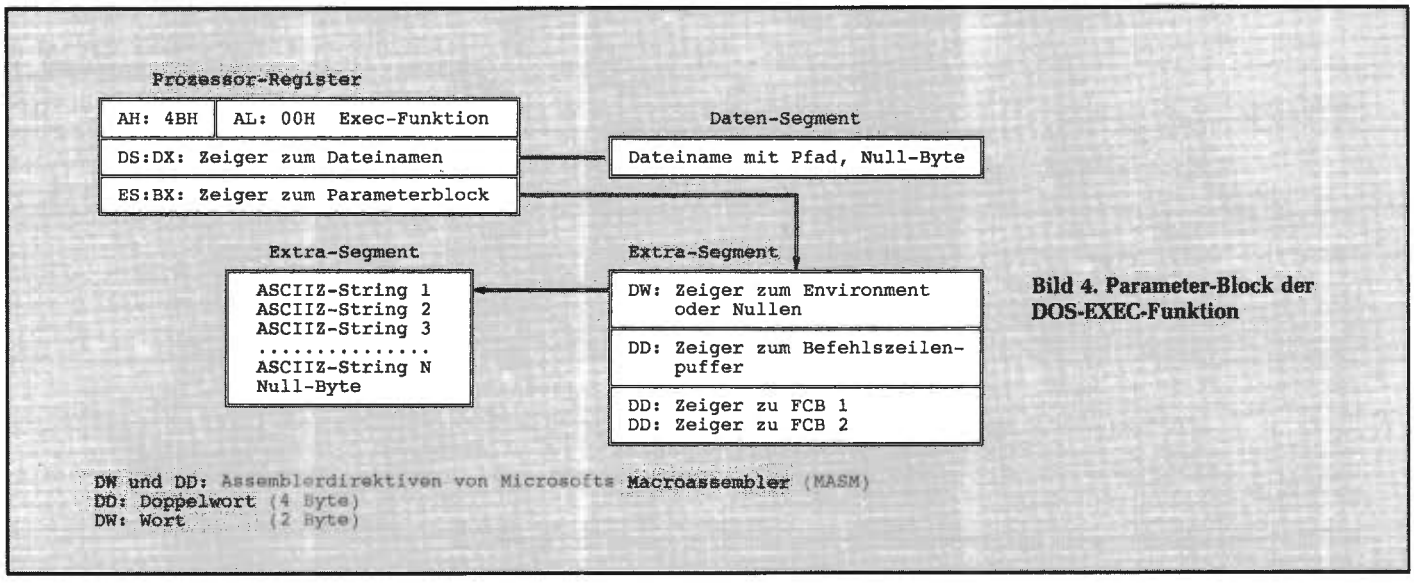


Bild 4. Parameter-Block der DOS-EXEC-Funktion

Peter Kuhn

Das ist der IBM-Mikro-Kanal-Bus

Der neue Bus für die IBM-Personals

Seit dem Erscheinen des ersten Mikroprozessor-Busses, des Multibus von Intel, sind gut und gerne 10 Jahre vergangen. Während dieser Zeit sind viele weitere Bussysteme spezifiziert worden, man denke nur an die Namen VME, Multibus II, PC I/O Bus, AMS, SMP, ECB oder an den relativ neuen NUBUS, der in der Architektur des Macintosh 2 von Apple seinen Platz gefunden hat.

Seitdem allerdings IBM Ende April eine neue Familie von Personalcomputern unter dem Namen PS/2 auf dem Markt eingeführt hat, verursacht ein ganz anderer Name allen Herstellern von Zusatzkarten für IBM-Rechner und Kompatible schlaflose Nächte. Mikro-Kanal heißt der (noch) geheimnis-umwitterte Bus, der in dieser neuen Rechnerfamilie eingesetzt wird. Im folgenden soll dieser neue Bus im Einzelnen unter die Lupe genommen werden.

Die neuen PCs

Für die Rechnerfamilie PS/2 wurden 4 Modelle angekündigt:
Modell 30 mit einem 8 MHz 8086, Modell 50 und Modell 60 mit einem 10 MHz 80286, sowie Modell 80 mit einem 16/20 MHz 80386. Während im Modell 30 noch der alte IBM PC I/O Bus eingesetzt wird, findet man im Modell 50, Modell 60 und Modell 80 den neuen Mikro-Kanal-Bus. Auf den System-Boards dieser Rechner wurde ein Großteil der Peripherie integriert, der bei bisherigen PCs noch als Zusatzkarten hinzugefügt werden mußte. So besteht das System-Board der Modelle 50/60 aus folgenden Einheiten:

- 10-MHz-Mikroprozessor 80286
- Fassung für 80287
- 8-Kanal DMA-Kontroller
- 16 Interruptebenen
- 128 KByte ROM
- 1 MByte RAM
- CMOS RAM für Kalender und Uhr
- Integrierter Video-Graphic-Kontroller (VGA)
- Parallel-Port
- Serial-Port
- Audio-Subsystem
- Keyboard-Kontroller
- Disketten-Kontroller
- Vier Mikro-Kanal-Steckplätze beim Modell 50, acht beim Modell 60, jeweils einer davon für den Winchester-Kontroller, so daß drei bzw. sieben freie Steckplätze für weitere Peripherie zur Verfügung stehen.

Im Gegensatz zu den bisherigen PCs von IBM sind auf dem System-Board kaum mehr diskrete Peripheriebausteine enthalten. Stattdessen machen mehrere ASICs, die erst einmal „geknackt" werden müssen, den Clone-Herstellern das Leben schwer. Die Belegung der System-Board-I/O-Adressen zeigt *Bild 1*. Sie ist bei den Peripheriekomponenten, deren Funktion sich nicht geändert hat, identisch mit den I/O-Adressen der bisherigen PCs.

HEX ADRESSEN	BAUSTEIN
0000 - 001F	DMA Kontroller
0020,0021	Interrupt Kontroller 1, 8259A
0040,0042,0043,0044,0047	System Timer
0060	Tastatur, AUX device
0061	System Kontrollport B
0064	Tastatur, AUX device
0070,0071	CMOS Uhr und NMI Maskierung
0074,0075,0076	Reserviert
0081,0082,0083,0087	DMA Page Register (0 - 3)
0089,008A,008B,008F	DMA Page Register (4 - 7)
0090	Kontrollport für zentralen Arbiter
0091	Card Selected Feedback
0092	System Kontrollport A
0093	Reserviert
0094	System Board Setup
0096,0097	POS, Adapterauswahl
00A0 - 00A1	Interrupt Kontroller 2, 8259A
00C0 - 00CF	DMA Kontroller
00F0 - 00FF	80287 Mathematischer Coprocessor
0100 - 0107	POS Register
0278 - 027B	Parallel Port 3
02F8 - 02FF	Serial Port 2
0378 - 037B	Parallel Port 2
03BC - 03BF	Parallel Port 1
03B4,03B5,03BA,03C0 - 03C5	Video Subsystem
03CE,03CF,03D4,03D5,03DA	Video Subsystem
03C6 - 03C9	Video DAC
03F0 - 03F7	Disketten Kontroller
03F8 - 03FF	Serial Port 1

Bild 1. Die I/O-Adressen des System-Boards

Die Mikro-Kanal-Architektur

Im Gegensatz zu dem bisherigen PC-I/O-Bus, der wie der Name schon sagt nur ein einfacher I/O-Erweiterungsbus des System-Boards und nur mit Klimmzügen multimasterfähig war, besitzt der Mikro-Kanal-Bus einige wesentliche neue Eigenschaften, die in etwas abgeänderter Form von Multibus II oder NUBUS schon bekannt waren:

- Multimaster-Betrieb mit bis zu 16 Busteilnehmern und zentraler/dezentraler Buszuteilung (Arbitration).
- Konfigurieren von Peripheriebaugruppen nach dem sog. POS-(Programmable Option Select) Verfahren per Software.
- DMA- und Burst-Betrieb (Blockmode)
- Fehlererkennungs- und Behebungsmechanismen.

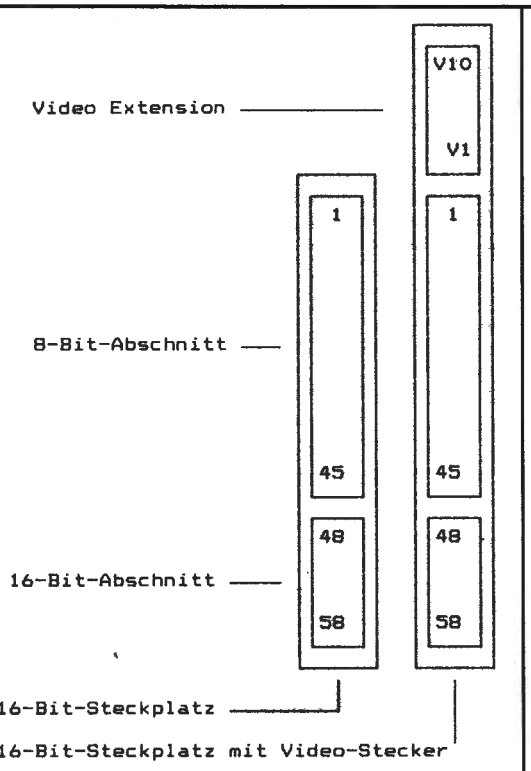


Bild 2. Die Mikro-Kanal-Slots

– Verbesserte Störstrahlungseigenschaften durch verteilte Masse und Versorgungsspannungsleitungen.

Ansonsten setzt sich der Mikro-Kanal-Bus aus mehren Teilbussystemen zusammen, einem Adreßbus, einem Datenbus, einem Kontrollbus, einem Bus für die Arbitration sowie mehreren zusätzlichen Signalen. Der Adreß- und Datenbus erlaubt beim Modell 50/60 die Adressierung von 16 MByte mit 8- oder 16-Bit-Speicherzugriffen. Beim Modell 80 ist der Zugriff auf 32 Bit erweitert, wegen des 80386 Mikroprozessors. Der I/O Adreßbereich ist wie bisher auf 64 KByte festgelegt. Ein serielles DMA-Protokoll unterstützt 8 DMA-Kanäle, die Anzahl der maximalen Interrupt-Ebenen beträgt 16.

Genau wie beim „alten PC" ist beim Mikro-Kanal eine Unterteilung des Steckers in eine 8-Bit-Standardbelegung und eine 16-Bit-Erweiterung vorgesehen. Damit können auch einfache, billige 8-Bit-Baugruppen als Steckkarten verwendet werden. Weiterhin ist jeweils ein Steckplatz im System für eine zusätzliche Grafikkarte um 20 Signale erweitert worden (*Bild 2*).

Der Bus: asynchron

Der Mikro-Kanal-Bus ist ein asynchroner Bus, d.h. sämtliche Bussignale kön-

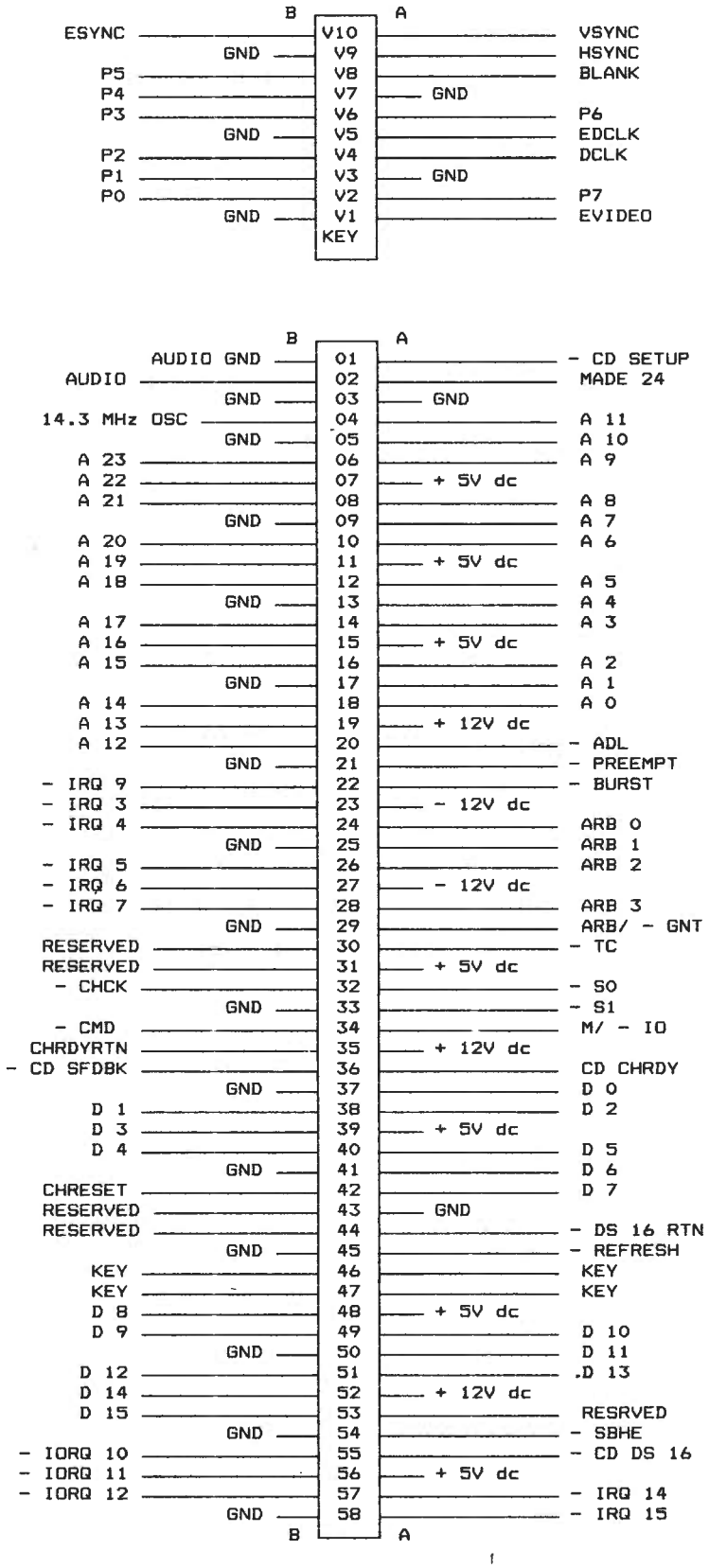


Bild 3. Die Pin-Belegung

nen sich zu beliebigen Zeiten ändern, – im Gegensatz zu einem synchronen Bussystem, wo Signalübergänge nur von einem zentralen Taktgeber gesteuert stattfinden können. Aufgrund seiner Architektur, die im wesentlichen die Signale des 80286 direkt auf den Bus legt, ist mit Slave-Baugruppen, wenn sie keine Wartezyklen einfügen, eine maximale Busbandbreite von 10 MByte/s erreichbar. Dieses entspricht der Busbandbreite eines 10-MHz-80286, d.h. es geht auch bei einer Datenübertragung über den externen Bus praktisch keine Zeit verloren.

Genau wie der „alte" PC-I/O- Bus ist der Mikro-Kanal-Bus mit direkten Steckplätzen (Edge Connector) versehen. Die Erweiterungskarten werden einfach in die Schlitze des System-Boards eingesteckt. Was werden wohl die Verfechter der indirekt gesteckten Europakarten dazu sagen ?? Fast überflüssig zu erwähnen, daß die bisher erhältlichen Erweiterungskarten natürlich nicht mehr mit dem Mikro-Kanal zusammenpassen. Und hier liegt auch die große Chance für alle Clone-Hersteller. Wer wird wohl der erste sein, der einen IBM kompatiblen Rechner sowohl mit PC-I/O- Slots als auch mit Mikro-Kanal-Slots auf den Markt bringt?

Die Mikro-Kanal-Signale

Bild 3 zeigt die Pinbelegung des Mikro-Kanal-Busses. Die *Tabelle* zeigt die Signalarten und einige Optionen des Busses. Im Folgenden soll etwas näher auf die Funktion der einzelnen Bussignale eingegangen werden:

A0...A23:

Adreßbits A0 (LSB)...A23 (MSB) für die Adressierung des Speicher- und I/O-Adreßraumes. Bei Speicherzugriffen werden alle Adreßbits zur Adressierung bis 16 MByte benutzt, für den I/O-Adreßraum (64 KByte) werden lediglich die Leitungen A0...A15 verwendet, die auf Slavebaugruppen vollständig decodiert werden müssen, um doppelte Adressierung mehrerer Peripheriebaugruppen zu verhindern. Die Adreß-Signale werden vom Systemprozessor (80286) generiert und bleiben „ungelatched". Sie können entweder mit der mit der steigenden Flanke von $\overline{ADL}$ oder mit der fallenden Flanke von $\overline{CMD}$ in einen Auffangspeicher übernommen werden. Um einen einwandfreien Multimaster-Betrieb zu gewährleisten, müssen A0...A23 mit Tri-State-Treibern versehen werden.

D0...D15:

Daten-Bits D0 (LSB)...D15 (MSB). Gültig für die Systemplatine und alle Slave-Baugruppen. Alle 8-Bit-Slave-Baugruppen benutzen nur D0...D7. Während der Lesezyklen müssen die Daten nach der fallenden Flanke von $\overline{CMD}$ bis nach der steigenden Flanke von $\overline{CMD}$ gültig sein. Bei Schreibzyklen müssen die Daten für die gesamte Dauer von $\overline{CMD}$ anliegen. Alle Datenleitungen müssen mit Tri-State-Treibern versehen werden.

$\overline{ADL}$:

Das Adreß-Decode-Latch-Signal entspricht dem invertierten ALE des 80286 und wird zum Latchen der Adreß- und Statusleitungen benutzt. Es wird empfohlen, transparente Latches auf den Slavebaugruppen zu verwenden, und diese entweder mit der steigenden Flanke von $\overline{ADL}$ oder der fallenden Flanke von $\overline{CMD}$ zu latchen. $\overline{ADL}$ muß mit einem Tri-State-Treiber versehen werden.

$\overline{CMD}$:

Das Command-Signal zeigt an, wann gültige Daten auf dem Bus anliegen, siehe auch D0...D15. Die Rückflanke von $\overline{CMD}$ beendet jeweils einen Buszyklus.

M/$\overline{IO}$:

Dieses Signal unterscheidet Speicher- (wenn positiv) und I/O-Zugriffe (wenn negativ).

$\overline{S0}$, $\overline{S1}$:

Diese Signale entsprechen in ihrer Bedeutung den Statusleitungen des 80286. Lediglich die Decodierung von INTA und HALT des 80286 wurde nicht auf den Bus gelegt. Für die Decodierung der Statussignale , die den Beginn eines Mikro-Kanal-Zyklus anzeigen, gilt die Zuordnung nach *Bild 4*.
Decodierte Signale müssen aus den zwischengespeicherten Statusleitungen ($\overline{S0}$, $\overline{S1}$), M-$\overline{IO}$ und $\overline{CMD}$ gebildet werden.

Tabelle: Die Signalarten am 16-Bit-Kanal

Signal Name	System Logic	DMA Cntlr	Intelligenter Bus-Cntlr	DMA Slave	MEM Slave	I/O Slave	Driver Typ
	D/R	D/R	D/R	D/R	D/R	D/R	(Signal Gruppe)
A (0–23)	D/–	D/–	D/–	–/R	–/R	–/R	TS (1)
D (0–15)	D/R	D/R	D/R	D/R	D/R	D/R	TS (2)
–ADL	D/–	D/–	D/–	–/O	–/O	–/O	TS (1)
–CD DS 16 (n)	–/R	–/R	–/R	#/–	#/–	#/–	TP (3)
–DS 16 RTN	D/–	–/R	–/R	–/–	–/–	–/–	BD (4)
–SBHE	D/–	D/–	D/–	–/#	–/#	–/#	TS (1)
MADE 24	D/–	D/–	D/–	–/–	–/R	–/–	TS (1)
M/–IO	D/–	D/–	D/–	–/R	–/R	–/R	TS (1)
–S0, –S1	D/–	D/–	D/–	–/R	–/R	–/R	TS (1)
–CMD	D/–	D/–	D/–	–/R	–/R	–/R	TS (1)
–CD SFD8K (n)	–/R	–/–	–/–	D/–	D/–	D/–	TP (3)
CD CHRDY (n)	O/R	–/–	–/–	O/–	O/–	O/–	TP (3)
CHRDYRTN	D/–	–/R	–/R	–/–	–/–	–/–	BD (4)
ARB (0–3)	O/R	–/R	D/R	D/R	–/–	–/–	OC (5)
–BURST	O/R	D/R	D/–	#/–	–/–	–/–	OC (5)
–PREEMPT	O/R	–/–	D/R	D/#	–/–	–/–	OC (5)
ARB/–GNT	D/–	–/R	–/R	–/R	–/–	–/–	BD (4)
–TC	–/–	D/–	–/–	–/O	–/–	–/–	TS (1)
–IRQ (*)	–/R	O/–	O/–	O/–	–/–	O/–	OC (6)
–CD SETUP (n)	D/–	–/#	–/R	–/R	–/R	–/R	TP (7)
–CHCK	–/R	#/–	D/O	D/–	D/–	D/–	OC (6)
–REFRESH	D/R	–/O	–/O	–/O	–/R	–/O	TS (1)
OSC	D/–	–/O	–/O	–/O	–/O	–/O	CD (7)
CH RESET	D/–	–/O	–/R	–/R	–/R	–/R	BD (4)

KEY

D = Drive Enabled
O = Optional
R = Receive Enabled
– = Not Implemented
= Some are Required

OC = Open Collector
TS = Tri-State
TP = Totem-Ple
BD = Bus Driver
CD = Clock Driver

* IRQ (9–12, 14, 15, 3–7)

Alle Slavebaugruppen müssen die Statussignale vollständig decodieren. Weiterhin müssen sämtliche Adreß-Signale schon vor der Aktivierung von $\overline{S0}$ gültig sein. Während decodierter Lesezyklen (Memory- oder I/O-Baugruppe) müssen die Daten nach der fallenden Flanke von $\overline{CMD}$ bis nach der steigenden Flanke von $\overline{CMD}$ gültig sein. Bei Schreibzyklen müssen die Daten für die gesamte Dauer von $\overline{CMD}$ anliegen.

M-IO/	SO/	S1/	Bedeutung (decodiertes Signal)
0	0	0	Reserviert
0	0	1	IO Write
0	1	0	IO Read
0	1	1	Reserviert
1	0	0	Reserviert
1	0	1	Memory Write
1	1	0	Memory Read
1	1	1	Reserviert

Bild 4. Die Signale M/$\overline{IO}$,$\overline{S0}$ und $\overline{S1}$

$\overline{SBHE}$:

Das System-Byte-High-Enable-Signal zeigt an, daß die Daten D8...D15 (high Byte) des Datenbusses gültig sind. A0 zeigt hingegen an, daß die Daten D0...D7 (low Byte) gültig sind.

MADE 24:

Memory Adress Enable, aktiv high, zeigt an, daß ein Speicherzugriff mit einer Adresse < 16 MByte durchgeführt wird. Dieses Signal muß in Hinsicht auf die 32-Bit-Generation mit dem 80386, der ja 4 GByte physikalisch adressieren kann, von den Adreß-Decodern auf allen Speicherboards mitdecodiert werden.

$\overline{CD\ DS16}$ (n):

Card Data Size 16 ist ein Signal, das von allen 16-Bit-Speicher-, I/O- oder DMA-Slave-Baugruppen geliefert werden muß. 8-Bit-Slaves dürfen diese Leitung nicht aktivieren. Das (n) bedeutet, daß jeder Steckplatz im Mikro-Kanal eine eigene $\overline{CD\text{-}DS16}$-Leitung besitzt, die alle zu dem System-Board geführt werden. Hier werden alle n $\overline{CD\text{-}DS16}$-Leitungen negativ verodert und bilden das Signal

$\overline{DS16\ RTN}$:

Data Size 16 Return. Dieses Signal wird aktiv, sobald eine der Slave-Baugruppen seine $\overline{CD\text{-}DS16}$-Leitung aktiviert. Andere Busmaster als das System-Board können über diese Leitung die aktuelle Datenbreite beobachten. Beide DS16 Signale wurden im Hinblick auf den 80386 vorgesehen, der ja auch einen Pin mit der Bezeichnung BS16 (Bus Size 16) besitzt, über den die Datenbreite (16 oder 32 Bit) des 80386 eingestellt werden kann.

$\overline{CD\ SETUP}$ (n):

Card Setup (n) ist eine Gruppe von Signalen die auf dem System-Board erzeugt werden und zum Zweck der automatischen Konfiguration externer Adapter (POS) einzeln zu allen Steckplätzen geführt werden. Auf dem einzelnen Adapter wird $\overline{CD\ SETUP}$ (n) über ein UND-Gatter mit dem I/O-Read- und dem I/O-Write-Signal verknüpft, sodaß die Konfigurationsdaten der einzelnen Adapter nur dann verarbeitet werden können, wenn $\overline{CD\ SETUP}$ aktiv ist. Diese Leitung wird softwaremäßig durch Beschreiben von I/O-Adresse 96H (Channel Position Select Register) mit der jeweiligen Adapterposition (0...7) auf dem System-Board aktiviert bzw. deaktiviert.

$\overline{CD\ SFDBK}$ (n):

Card Select Feedback ist eine Gruppe von Signalen, die bei jeder Adressierung einer Slavebaugruppe durch den System-Mikroprozessor aktiviert wird. Quasi als positive Bestätigung, daß die Karte, deren Speicher- oder I/O-Bereich gerade angesprochen wird, sich wirklich im System befindet. Diese für Diagnosezwecke vorhandenen Signalen dürfen nicht während der automatischen Konfiguration, angezeigt durch $\overline{CD\ SETUP}$ aktiviert werden.

CD CHRDY (n):

Die Gruppe der Channel-Ready-Signale entspricht der normalen Ready-Logik, wie sie von fast allen Mikroprozessoren bekannt ist. Solange dieses Signal aktiv (positiv) ist, werden bei Zugriffen auf externe Baugruppen keine Wartezyklen eingefügt. Sobald diese Leitung auf low gezogen wird, verlängert der System-Mikroprozessor den gerade laufenden Buszyklus solange, bis CH CHRDY(n) wieder positiv ist. Die maximale Zeit hierfür ist mit 3 µs spezifiziert. Alle CD-CHRDY-Signale der einzelnen Adapter werden auf der System-Platine zusammengeführt und bilden dort das Signal

CHRDYRTN:

Channel Ready Return. Es ist die positive UND-Verknüpfung aller CD-CHRDY-(n)-Signale der einzelnen Adapter und erlaubt es somit anderen Busmastern, die Ready-Information zu prüfen und die Buszyklen gegebenenfalls zu verlängern.

$\overline{PREEMT}$:

Die Leitung Preempt einer intelligenten Busbaugruppe (DMA oder I/O) zeigt an, daß sie sich im Moment aktiv um einen Buszugriff bemüht. Hierdurch wird ein Arbitrationszyklus eingeleitet.

ARB0...ARB3:

Anhand dieser Signale, die 16 Prioritätsebenen für Multimaster-Betrieb definieren, wird die jeweilige Buszuteilung (Arbitration) bei mehreren Busmastern geregelt. Baugruppen mit ARB0...ARB3 = 0 haben die höchste Priorität, solche mit 0FH haben die niedrigste Buspriorität. Diese Signale werden von dezentralen Arbitern auf den jeweiligen Adapterbaugruppen erzeugt.

ARB /$\overline{GNT}$:

Arbitration $\overline{Grant}$. Dieses Signal, das von einem zentralen Arbiter erzeugt wird, signalisiert einen gerade laufenden Arbitrationszyklus (ARB). Mit einem negativen Pegel ($\overline{GNT}$) zeigt der zentrale Arbiter dem jeweiligen Busadapter mit der höchsten Priorität an, daß er gewonnen hat.

$\overline{BURST}$:

Mit der Aktivierung des Burst-Signals zeigt ein Slave am Bus dem zentralen Arbiter an, daß er einen Block von Daten in einem sogenannten Burst-Zyklus übertragen will. Diese Leitung ist allen Steckplätzen gemeinsam und wird von Open-Kollektor-Treibern aktiviert.

$\overline{TC}$:

Die Leitung Terminal Count zeigt das Ende eines DMA-Buszyklus an und wird vom DMA-Master aktiviert.

IRQ9...IRQ12
IRQ14...IRQ15
IRQ3...IRQ7

Interrupt-Request-Leitungen für alle Busadapter zum zentralen System-Board. Sie sind in ihrer Priorität festgelegt und nicht veränderbar. Interrupt 9 hat die höchste, Interrupt 7 die niedrigste Priorität. Die effektive Interrupt-Priorität lautet also: IRQ (9...12, 14, 15, 3...7).

Alle Interrupts müssen sogenannte. „Level-Sensitiv"-Interrupt-Signale sein, da der Interrupt-Kontroller nicht auf „Edge-sensitive"-Signale reagiert und sich auch per Software nicht dazu bringen läßt, solche Signale zu erkennen.

CHCK:

Channel Check ist eine Open-Kollektor-Leitung aller Adapterbaugruppen. Über sie werden Fehler angezeigt, die einen sofortigen Systemabsturz zur Folge haben können (z.B. Parity Error). CHCK muß solange low bleiben, bis es vom CHCK Interrupt Handler zurückgesetzt wird.

AUDIO:

Analoge Audio-Leitung zwischen den einzelnen Adaptern und dem Audio-System-Ausgang. Der Frequenzbereich ist von 50 Hz bis 10 KHz spezifiziert und hat eine maximale Amplitude von 2.5 Volt (Vss).

AUDIO GND:

Audio-Masse, getrennt von allen digitalen Masseleitungen.

OSC:

Oszillator. Zentraler Takt mit 14.31818 MHz.

CHRESET:

Channel-Reset-Signal mit dem alle Adapter nach dem Power Up in den Grundzustand versetzt werden. Dieses Signal kann auch unter Softwarekontrolle des System-Boards aktiviert werden.

REFRESH:

Refresh-Signal des System-Boards um anzuzeigen, daß gerade ein Refresh-Zyklus des dynamischen RAMs durchgeführt wird. Zusammen mit dieser Leitung werden die Adreßleitungen A0...A8 aktiviert. Dieses Signal muß sich nicht an einen festen Zeitrahmen halten und sollte deswegen nicht zur Ableitung irgenwelcher Timings verwendet werden.

Die folgenden Signale sind als sogenannte Auxiliary Video Extension als Erweiterung des normalen Adapterstekkers an je einem Steckplatz des Mikro-Kanal-Busses untergebracht:

VSYNC:

Vertical-Sync-Signal

HSYNC:

Horizontal-Sync-Signal

BLANK:

Blank-Signal, um den Video-DAC im Zeilen/Bildrücklauf auszublenden.

P0...P7:

Diese Signale enthalten die digitale Videoinformation und steuern die Adreßeingänge für den Video-DAC.

DCLK:

Digitaler Video-Takt, um die digitale Videoinformation in den DAC zu speichern. Falls eine zusätzliche Videokarte benutzt wird, kann diese auch als Clock Input (EXTCLK) für den VGA-Adapter auf dem System-Board benutzt werden. Dabei müssen aber auch P0...P7 von der zusätzlichen Videokarte geliefert werden.

ESYNC:

Output-Enable-Signal für VSYNC, HSYNC und BLANK. Wenn dieses Signal high (+5V) ist, werden VSYNC, HSYNC und BLANK vom VGA auf dem System-Board geliefert, wenn low (0V) liefert der zusätzliche Videoadapter die Synchronsignale für den Bildschirm.

EVIDEO:

Output-Enable-Signal für die Puffer, die P0...P7 ansteuern. Wenn high (+5V), werden P0 – P7 vom VGA des System-Boards geliefert, wenn low (0V) liefert

Hardware	Software
1. Interrupt-Bedingung setzt Interrupt-Leitung (n) auf aktiv low und setzt zugehöriges Interrupt Latch.	
2. Interrupt Controller (8259A) auf dem System Board unterbricht die 80286 CPU.	
	3. Beginn des Interrupt Handlers, der zu dem ausgelösten Interrupt gehört.
	4. Der Interrupt Handler liest der Reihe nach (beginnend mit dem ersten Adapter des Mikro-Kanals) die Interrupt Latches aller Baugruppen, die mit dem ausgelösten Interrupt verbunden sind. Sobald er das gesetzte Latch gefunden hat, verzweigt er in die entsprechende Interrupt Service Routine (ISR).
	5. Die ISR setzt das Interrupt Latch zurück.
6. Die Hardware setzt das Interrupt Latch und damit auch den Interrupt zurück.	
	7. Die ISR beendet ihr Programm mit einem End of Interrupt (EOI) zum 8259A.
8. Der 8259A wir intern zurückgesetzt.	
9. Wenn ein weiterer Interrupt anliegt (IRQ aktiv von einem anderem Adapter), unterbricht der 8259A den 80286 erneut und das Ganze beginnt von vorne.	

Bild 5. Der Ablauf bei Interrupts

der zusätzliche Videoadapter die Signale P0...P7.

EDCLK:

Output Enable für den DCLK-Puffer. Wenn high (+5V) wird der DCLK vom VGA des System-Boards geliefert, wenn low (0V) muß DCLK vom zusätzlichen Videoadapter erzeugt und dem EXTCLK Eingang des VGA zugeführt werden.

Wie aus Bild 4 zu sehen ist, sind zwischen jeweils drei Signal-Leitungen eine Masse- oder Versorgungsspannungsleitung vorgesehen. Hiermit soll dem Übersprechen zwischen benachbarten Signalen vorgebeugt werden.

Das Mikro-Kanal-Interrupt-System

Wie beschrieben sind die Interrupts der Mikro-Kanal-Architektur als sog. „Level Sensitive Interrupts" mit aktiv-low-Pegel definiert, um eine höhere Störsicherheit zu erhalten. Hierdurch wird auch das Auftreten sogenannter „Spourious" Interrupts (das sind Interrupts, die durch irgendwelche Umgebungsbedingungen auftreten können) völlig vermieden. Diese „Spourious" Interrupts lösten beim 8259A-Interrupt-Kontroller den Interrupt 7 aus und mußten in einer speziellen Interrupt-Routine abgefangen werden. Ein direktes Programmieren des 8259A-Interrupt-Kontrollers auf „Edge-Sensitive"-Betrieb, ist nicht möglich, da dies von der System-Board-Hardware verhindert wird. Um die Anzahl der durch die Hardware festgelegten Interrupts zu erweitern, ist es möglich, mehrere Interrupt-Quellen gemeinsam über Open-Kollektor-Treiber physikalisch auf eine Interrupt-Leitung zu legen. Da dieses „Interrupt Sharing" bei der Mikro-Kanal-Architektur vorgesehen ist, müssen alle Interrupt-Leitungen auf den Adapter-Boards mit Latches versehen sein, die per Software vom System-Board gelesen bzw. zurückgesetzt werden können. Hierbei muß allerdings der Interrupt-Handler alle Adapter, die eine Interrupt-Leitung gemeinsam benutzen abfragen, um die eigentliche Interruptquelle festzustellen. Da hierdurch die Interrupt-Antwort-Zeit erheblich verlängert werden kann, sollte dieses Verfahren nur im Notfall angewendet werden. *Bild 5* zeigt, wie Hard- und Software bei Eintreten eines Interrupts beim Interrupt Sharing zusammenspielen.

Teil 2 dieses Artikels wird im Einzelnen auf folgende Eigenschaften der Mikro-Kanal-Architektur eingehen:

- *Automatische Konfiguration (POS)*
- *Arbitration bei Multimaster-Betrieb*
- *DMA und Burst Mode*
- *Zeitverhalten (Timings)*

HP-Expo '87

Im Mai 1987 veranstaltete die Firma Hewlett und Packard in Böblingen eine Ausstellung in den eigenen Firmenräumen, die HP-EXPO '87. Geladen waren qualifizierte HP-Kunden und solche, die es werden sollten. Gezeigt wurden die technischen Computersysteme von HP und deren Einsatzfälle in der Industrie. Während der Expo wurden Fachvortäge gehalten. Eine kleine Auswahl aus den Themen: „CAD-Trends", „Mikroprozessor-Entwicklung", „Fertigungsautomatisierung: Herausforderung und HP-Erfahrung", „Netzwerke, Basis für die Fabrik der Zukunft", „CAQ - Qualitätssicherung" - und vieles mehr. Zusätzlich wurden Werksführungen angeboten, auf welchen man zum Beispiel anhand der Produktion des HP-Systems 9000, Serie 300 eine Fertigung auf Abruf studieren konnte. Ein anderes Beispiel war die HP-eigene Blechfertigung, die zur Zeit auf CAM-Verfahren umgestellt wird. In den Ausstellungsräumen hatten Partnerfirmen Gelegenheit, ihre Komplettlösungen anzubieten. Darunter waren so große Firmen wie Siemens, Dornier und NEC. Das schwerste Exponat war ein Rechnergesteuertes Universal-Meßzenrum der Firma Carl Zeiss, das tonnenschwere, luftgelagerte Granit-Blöcke zur Stabilisierung und Positionierung der Meßtaster und der abzutastenden Objekte benutzte. *Ro.*

Mit DES Daten schützen

Die Firma Krypto-Soft, Bergisch Gladbach, hat ein Verschlüsselungspaket entwickelt, das Dateien schützt. Nach dem DES-Algorithmus arbeitend (Data Encryption Standard), verschlüsselt das Programm Krypto-Star Byte für Byte durch Verrechnung mit einem 64 Bit langen Grundschlüssel. Darüberhinaus hängt das Verschlüsselungsergebnis noch von einem zweiten 64-Bit-Wort ab, das den Startwert bildet. Nach den bisherigen Erkenntnissen ist die Verschlüsselung nach DES sicher. Das bedeutet unter Kryptologen, daß der Aufwand, wenn man rein durch kombinatorische Verfahren in den verschlüsselten Text einbrechen will, so hoch ist, daß der Einbruch auch mit den schnellsten derzeit existierenden Rechnern nicht innerhalb überschaubarer Zeiten gelingt.

Während zum Beispiel die Euroscheck-Kartenleser intern mit ICs per Hardware nach DES arbeiten, ist Krypto-Star ein reines Software-Produkt. Es läuft unter MS-DOS, RTOS-UH und Atari-GEM. Zu Krypto-Star gibt es noch ein Programm namens Krypto-Convert, das von Krypto-Star verschlüsselte Daten über die DFÜ-Schnittstelle des entsprechenden Computers ausgeben kann. Mit den Krypto-Programmen kann man jetzt sehr preiswert alles Wichtige im Computer schützen, denn Krypto-Star kostet nur 198 DM.

Nach Unterlagen von Krypto-Soft

Ingrid Trommer

Einführung in Unix

Teil 2: Laufen lernen ...

Unix kennt eine stattliche Anzahl von Befehlen. Unsere Autorin, Mitarbeiterin von Hewlett-Packard, erklärt in dieser Folge wichtige Befehle des Dateisystems. MS-DOS-Anwendern werden einige Befehle sehr bekannt vorkommen.

Zunächst eine Anmerkung: Unix unterscheidet zwischen Groß- und Kleinschreibung. Befehle werden grundsätzlich klein geschrieben. Bei Dateinamen hat der Benutzer die freie Wahl. Hält man sich an Konventionen, so gilt auch hier der Grundsatz Kleinschreibung.
Um mit dem Rechner kommunizieren zu können, muß der Rechner den Benutzer erst einmal kennen. Ein neuer Benutzer wird vom Systemadministrator – auch Superuser genannt – dem Rechner bekannt gemacht. Der Systemadministrator hat den Login-Namen root und darf auf dem System fast alles tun. Er kann z. B. trotz fehlender Zugriffsberechtigungen alle Dateien löschen. Vom Systemadministrator werden – je nach lokaler Installation und Hersteller – Login-Namen und manchmal auch Paßwörter vergeben. Der normale Ablauf beginnt mit der Aufforderung „login :". Hat der Benutzer mit dem ihm zugeteilten Namen richtig geantwortet, so gibt es zwei Möglichkeiten:

- Es wurde kein Paßwort vergeben, dann erscheint nach herstellerspezifischen Meldungen die Eingabeaufforderung (prompt). Für den Normalbenutzer ist als Prompt das Dollarzeichen üblich.
- Es wurde ein Paßwort vergeben, dann erscheint als nächste Ausgabe die Aufforderung „password :".

Im Gegensatz zu allen anderen Eingaben, gibt der Rechner bei der Eingabe des Paßwortes kein Bildschirm-Echo aus. Ist das Paßwort richtig eingetippt, geht es weiter wie oben.
War eine der Eingaben falsch, dann erscheint die Meldung:
login incorrect
Viel Erfolg beim nächsten Versuch!
Bild 1 zeigt die Login-Sequenz eines Benutzers, dem vom Systemadministrator kein Paßwort zugeteilt wurde, und der sich selbst ein Paßwort gibt. Dazu ruft er den Befehl „password" auf. Aus dem Beispiel geht hervor, daß das System keine Paßwörter erlaubt, die kürzer als sechs Zeichen sind. Eine Zahl oder ein Sonderzeichen muß immer enthalten sein. Hat das System das Paßwort akzeptiert, so verlangt es die Eingabe, um Schreibfehler auszuschließen, noch einmal. Dies gilt für Mehrbenutzer-Systeme, die bereits einige Zeit in Betrieb sind. Bei Einbenutzer-Systemen oder bei neu installierten Systemen variiert das Verhalten je nach Hersteller. Zwischen zwei Systemverhalten kann unterschieden werden:

- Das System kommt mit der Aufforderung „login :". Darauf antwortet der Benutzer mit root. Hier verhalten sich die einzelnen Implementierungen wiederum unterschiedlich. Es können verschiedene Meldungen zu den Urheberrechten angezeigt werden. Irgendwann sollte die Eingabeaufforderung (Prompt) für den Superuser (normalerweise „#") am Terminal erscheinen.
- Nach diversen Meldungen zeigt das System direkt den Superuser-Prompt an.

```
login: hugo
Erase set to Backspace
(c) Copyright 1983, 1984, 1985 Hewlett-Packard Co.
(c) Copyright 1979 The Regents of the University of Colorado, a body corporat
(c) Copyright 1979, 1980, 1983 The Regents of the University of California
(c) Copyright 1980, 1984 AT&T Technologies. All Rights Reserved.

Welcome to Hewlett-Packard System 9000 HP-UX

$ passwd
Changing password for hugo
New password: _____
Password is too short - must be at least 6 digits
New password: _____
Password must contain at least two alphabetic characters and
at least one numeric or special character.
New password: _____
Re-enter new password: _____
$
```

Bild 1. So präsentiert sich Unix gegenüber dem Anwender während der Login-Prozedur

Einfache Befehle

Der Anfänger sollte die ersten Gehversuche nicht als Superuser unternehmen. Es ist doch etwas umständlich, das System immer wieder neu zu installieren. Die Eingaben des Anwenders stellen wir im Text und in den Bildern unterstrichen dar, damit sie von den Antworten des Systems unterscheidbar sind.
Zunächst sollen Datum und Uhrzeit ausgegeben werden :

```
$ date
Monday January 5 15:21:05 MEZ
```

Wer arbeitet am System ?

```
$ who
ingrid console  Jan 5 14:22
peter tty01     Jan 5  8:12
hugo tty02      Jan 5  9:47
uli tty07       Jan 5  9:36
```

Die Ausgabe zeigt zunächst den Login-Namen des jeweiligen Benutzers. Darauf folgt der Namen des Terminals, so wie er in dem Directory /dev zu finden ist. Datum und Uhrzeit entsprechen dem Zeitpunkt des Einloggens.

Befehlsaufbau

Alle bis jetzt besprochenen Befehle waren insofern sehr einfach, da der Aufruf aus nur einem Wort bestand. Unter Unix ist es jedoch die Regel, daß ein Befehl Argumente hat. Außer dem Befehlsnamen können alle Argumente optional sein. Standardmäßig hat ein Befehl (Vorsicht, es gibt viele Ausnahmen!) das Format:
Befehlsname Option(en) Bitmuster Dateiname(n)

Die Optionen bestehen aus einzelnen Buchstaben und sind durch ein vorange-

```
DATE(1)

NAME
      date - print and set the date

SYNOPSIS
     date [ [yy]mmddhhmm ] [ +format ]

HP-UX COMPATIBILITY
     Level:     HP-UX/NUCLEUS

     Origin:    System V

     Native Language Support:
                8-bit data, customs, messages

DESCRIPTION
     If no argument is given, or if the argument begins with +,
     the current date and time are printed. Otherwise, the
     current data is set. yy is the last two digits of the year
     number; the first mm is the month number; dd is the day
     number in the month; hh is the hour number (24 hour system);
     the second mm is the minute number. The year, month, and
     day may be omitted, the current values being the defaults.
     For example:

     date 10080045
```

Bild 2. Ein Unix-System kann seine Befehle selbst erläutern

stelltes „-" oder manchmal auch durch „+" gekennzeichnet. Sie beeinflussen das Verhalten eines Befehls. Ein Bitmuster (normalerweise eine Buchstabenfolge) dient als Eingabe für einen Befehl. Im einfachsten Fall ist die Buchstabenfolge selbst die Eingabe. Bitmuster werden normalerweise bei den Befehlen benötigt, die zur Textmanipulations-Gruppe gehören (Erläuterung in einer der nächsten Folgen). Unter der Option Dateinamen stehen Dateien, die als Eingabe, Ausgabe oder beides dienen. Man darf allerdings nicht denselben Namen, sowohl für die Eingabe-, als auch die Ausgabedatei benutzen. Die Shell würde in einem solchen Fall die Eingabedatei zuweisen, den Namen der Ausgabedatei erkennen, und die Daten dort löschen.

Unix-Referenz-Manual im System!

Die meisten Unix-Systeme haben auf dem Sekundärspeicher das Unix-Referenz-Manual abgespeichert. Ist sich der Anwender über die Syntax eines Befehls nicht im Klaren, so genügt die Eingabe von „man date" und auf dem Terminal erscheint die in *Bild 2* dargestellte Ausgabe.
Das ist sicherlich eine gute Hilfe, allerdings nur dann, wenn der Anwender genau weiß, wie der Befehl geschrieben wird. Kennt er die Schreibweise nicht, so muß er die gedruckte Dokumentation zu Rate ziehen. Am Ende jeden Bandes ist auf den Seiten, die mit „Permuted Index" überschrieben sind, jeder Befehl aufgelistet und mit Schlagwörtern beschrieben.

Navigation durch das Dateisystem

Der grundsätzliche Aufbau des Dateisystems wurde bereits in der letzten Ausgabe beschrieben. Hier folgen die Befehle, die sich auf das Dateisystem beziehen.
Hat der Anwender die Einlog-Prozedur erfolgreich hinter sich gebracht, so befindet er sich in seinem Home-Directory. Auf den Befehl „pwd" (print working directory) gibt das System das momentane Dateiverzeichnis aus. Dies könnte z. B. lauten: /usr/hugo
Mit „cd" (change directory) läßt sich das Directory wechseln:
$ cd /usr (change directory/user)
$
Das System antwortet darauf mit dem nächsten Prompt. Will man den Erfolg überprüfen, so tut man das mit einem erneuten:
$ pwd (print working directory)
Darauf erfolgt die Antwort:
/usr
$
So reagiert das System (in den meisten Fällen) auf einen Schreibfehler:
$ pwg
pwg: not found
$
Noch eine weitere Befehlsfolge:
$ cd
$
$ pwd
/usr/hugo
$
Das System gibt keine positive Rückmeldung! Nach dem Befehl „cd/usr" erscheint schlicht der nächste Prompt.

Möchte man sehen, ob das System auch wirklich das Dateiverzeichnis gewechselt hat, so muß man sich mit der Ausgabe von „pwd" die Bestätigung holen. Für „cd/usr" hätte man auch „cd.." eingeben können, wie das in Kapitel 1 beschrieben wurde. Der Effekt wäre derselbe.
Eine Liste der Dateien oder Dateiverzeichnisse des momentanen Directory, gibt das System nach dem Befehl „ls" (list) aus. Das System zeigt alle vorhandenen Namen in alphabetischer Reihenfolge an. „ls" ist einer der Befehle, die sehr viele Optionen haben.
„ls-t" gibt die Namen nach der Reihenfolge ihrer letzten Modifikation aus.
„ls-l" gibt die Namen mit zusätzlicher Information aus.
Eine mögliche Ausgabe zeigt *Bild 3.*
An der ersten Stelle steht entweder ein „d" oder ein „-". „d" bedeutet hier Directory. Die nächsten neun Zeichen stehen für die neun Bit der gesetzten oder nicht gesetzten Zugriffsberechtigungen:
r: read (lesen)
w: write (schreiben)
x: execute (ausführen)

Diese Buchstabengruppe wiederholt sich dreimal
– 1. Eigentümer
– 2. Gruppe
– 3. Rest der Welt
(Wie in Folge 1 beschrieben.)

Die nächste Zahl ist die Anzahl der sogenannten Links einer Datei. Ist sie größer als eins, so bedeutet das, daß die Datei dem System unter mehr als einem Namen bekannt ist. Es folgen die Eintragungen des Namens des Datei-Eigentümers, die Anzahl der Zeichen, die Zeit der letzten Modifikation und zuletzt der Dateiname. Man kann Optionen zusammen angeben: z. B. „ls-lt". Gibt der Anwender keinen Directory-Namen an, so wird die Liste des momentanen Directories ausgegeben. Selbstverständlich kann auch der Namen eines Directories angegeben werden, um dessen Liste zu sehen. Will man den Inhalt einer Datei sehen, so lautet der Befehl dazu „cat" (concatenate = zusammenfügen).
Der Name wurde deswegen verwendet, weil mit „cat" nicht nur den Inhalt einer Datei, sondern auch von mehreren Dateien auf dem Terminal ausgegeben werden kann. Für den Anwender sieht es aus, als seien die Dateien aneinander angehängt worden. Der Befehl „cat daten1 daten2" gibt also den Inhalt von der Dateien daten1 und daten2 am Bildschirm so aus, als wenn eine Datei ausgegeben würde.

Als nächstes soll ein Dateiverzeichnis kreiert werden.

```
$ mkdir ohugo
$
```

„mkdir" (make directory) erzeugt ein Dateiverzeichnis im momentanen Directory. Bei diesem Beispiel wurde ein relativer Pfadnamen benutzt (relativ zum momentanen Directory). Alternativ kann der Anwender den absoluten Pfadnamen „mkdir /usr/hugo/ohugo" eingeben. Bei der absoluten Adressierung wird der Name ausgehend von Root (/), beschrieben. Danach folgt der Name jedes Unterdirectories, getrennt durch ein „/", bis man schließlich bei dem eigentlichen Namen angekommen ist.

```
$ ls -l
total 36
-rwxrwxrwx   1 hugo      users      13973  Jan 15 16:25 a.out
-rw-rw-rw-   1 hugo      users       1735  Jan 15 16:22 daten1
-rw-rw-rw-   1 hugo      users         29  Jan 15 16:22 daten2
-rw-rw-rw-   1 hugo      users         60  Jan 15 16:24 zahl.c
$
```

Bild 3. Auf den Befehl „ls -l" wird das Directory mit zusätzlichen Informationen ausgegeben

Nun sollen Dateien kopiert, umbenannt, und die gleiche Datei unter verschiedenen Namen dem System bekannt gemacht werden (Bild 3). Mit dem Befehl „cp" für copy, wird eine Datei physikalisch kopiert. Sie ist anschließend zweimal auf dem Sekundärspeicher vorhanden.

Bild 4 zeigt den Befehl in seiner einfachsten Form.

Aus dem Beispiel geht hervor, daß es jetzt eine Datei mehr gibt als vorher. Da daten1 nach daten3 kopiert wurde, ist die Anzahl der Zeichen (1735) in beiden Dateien gleich.

```
$ cp daten1 daten3
$ ls -l
total 40
-rwxrwxrwx   1 hugo      users      13973  Jan 15 16:25 a.out
-rw-rw-rw-   1 hugo      users       1735  Jan 15 16:22 daten1
-rw-rw-rw-   1 hugo      users         29  Jan 15 16:22 daten2
-rw-rw-rw-   1 hugo      users       1735  Jan 15 16:22 daten3
drwxrwxrwx   1 hugo      users          0  Jan 15 16:32 ohugo
-rw-rw-rw-   1 hugo      users         60  Jan 15 16:24 zahl.c
$
```

Bild 4. Eine Datei wird kopiert

Es können auch mehrere Dateien in ein Directory kopiert werden. Im folgenden Beispiel werden alle Dateien, deren Name mit der Buchstabenfolge „daten" anfangen, in das Directory „ohugo" kopiert. Den Erfolg der Aktion kontrolliert man mit „cd ohugo" und anschließender Ausgabe der Liste (*Bild 5*).

```
$ cp daten* ohugo
$ cd ohugo
$ ls -l
total 10
-rw-rw-rw-   1 hugo      users       1735  Jan 15 16:40 daten1
-rw-rw-rw-   1 hugo      users         29  Jan 15 16:40 daten2
-rw-rw-rw-   1 hugo      users       1735  Jan 15 16:41 daten3
$
```

Bild 5. Alle Dateien, deren Name mit „daten" anfangen, werden in das Directory „ohugo" kopiert

Der Befehl „mv" (move) benennt eine Datei um. Es soll im ursprünglichen Directory die Datei daten3 nach daten4 umbenannt werden (*Bild 6*). Das Wort „move" heißt bewegen. Jeder wird sich jetzt fragen, warum der Befehlsname nicht „rename" lautet. Da mit „mv" auch die Pfadnamen von Dateien verändert werden können, erscheint der Name „mv" wieder sinnvoll. Für den Anwender sieht es aus, als würden die Dateien von einem Dateiverzeichnis in das andere bewegt. Diese Anwendungsart von „mv" zeigt das Beispiel in *Bild 7*.

```
$ cd ..
$ pwd
/usr/hugo
$ mv daten3 daten4
$ ls -l
total 42
-rwxrwxrwx   1 hugo      users      13973  Jan 15 16:25 a.out
-rw-rw-rw-   1 hugo      users       1735  Jan 15 16:22 daten1
-rw-rw-rw-   1 hugo      users         29  Jan 15 16:22 daten2
-rw-rw-rw-   1 hugo      users       1735  Jan 15 16:22 daten4
drwxrwxrwx   1 hugo      users         72  Jan 15 16:32 ohugo
-rw-rw-rw-   1 hugo      users         60  Jan 15 16:24 zahl.c
$
```

Bild 6. Mit dem Befehl „mv" lassen sich Dateien umbenennen

Mit „mv" können auch Directories umbenannt werden (*Bild 8*).

Mit dem Befehl „ln" (link) kann eine Datei, die physikalisch nur einmal vorhanden ist, mit zwei verschiedenen Namen angesprochen werden. Dieser Befehl erscheint erst dann sinnvoll, wenn folgendes bekannt ist: Eine Shell-Prozedur (siehe nächste Folge) oder ein übersetztes Programm kann abfragen, mit welchem Namen sie (es) aufgerufen wurde. Sie (es) kann sich dann, abhängig vom Aufrufnamen, leicht unterschiedlich verhalten. Die Anwendung des „ln"-Befehls zeigt das Beispiel in *Bild 9*.

```
$ mv a.out zahl.c ohugo
$ ls -l
total 12
-rw-rw-rw-   1 hugo      users       1735  Jan 15 16:22 daten1
-rw-rw-rw-   1 hugo      users         29  Jan 15 16:22 daten2
-rw-rw-rw-   1 hugo      users       1735  Jan 15 16:39 daten4
drwxrwxrwx   1 hugo      users        120  Jan 15 17:02 ohugo
$ ls -l ohugo
total 40
-rwxrwxrwx   1 hugo      users      13973  Jan 15 16:25 a.out
-rw-rw-rw-   1 hugo      users       1735  Jan 15 16:40 daten1
-rw-rw-rw-   1 hugo      users         29  Jan 15 16:40 daten2
-rw-rw-rw-   1 hugo      users       1735  Jan 15 16:41 daten3
-rw-rw-rw-   1 hugo      users         60  Jan 15 16:24 zahl.c
$
```

Bild 7. So wird der Pfadname einer Datei verändert

Aus der zweiten Spalte ist zu ersehen, daß sich bei der Datei, die nun unter den Namen „daten4" und „daten5" erscheint, die Anzahl der Links um eins auf zwei erhöht hat.

Der Befehl zum Löschen einer Datei heißt „rm" (remove). Mit „rm daten4" wurde in *Bild 10* die Datei gelöscht, die mit „daten5" gelinkt war. Damit ver-

```
$ mv ohugo oha
$ ls -l
total 12
-rw-rw-rw-   1 hugo      users        1735  Jan 15 16:22 daten1
-rw-rw-rw-   1 hugo      users          29  Jan 15 16:22 daten2
-rw-rw-rw-   1 hugo      users        1735  Jan 15 16:39 daten4
drwxrwxrwx   1 hugo      users         120  Jan 15 17:02 oha
$
```

Bild 8. Ein Directory wird umbenannt

```
$ ln daten4 daten5
$ ls -l
total 16
-rw-rw-rw-   1 hugo      users        1735  Jan 15 16:22 daten1
-rw-rw-rw-   1 hugo      users          29  Jan 15 16:22 daten2
-rw-rw-rw-   2 hugo      users        1735  Jan 15 16:39 daten4
-rw-rw-rw-   2 hugo      users        1735  Jan 15 16:39 daten5
drwxrwxrwx   1 hugo      users         120  Jan 15 17:02 oha
$
```

Bild 9. Eine Datei, die nur einmal physikalisch vorhanden ist, kann unter zwei verschiedenen Namen angesprochen werden

```
$ rm daten4
$ ls -l
total 12
-rw-rw-rw-   1 hugo      users        1735  Jan 15 16:22 daten1
-rw-rw-rw-   1 hugo      users          29  Jan 15 16:22 daten2
-rw-rw-rw-   1 hugo      users        1735  Jan 15 16:39 daten5
drwxrwxrwx   1 hugo      users         120  Jan 15 17:02 oha
$
```

Bild 10. Der „rm"-Befehl dient zum Löschen von Dateien

schwindet der Name „daten4" aus der Liste, und der Link-Zähler bei „daten5" ist wieder eins. Will man ein Directory löschen, so heißt der Befehl dazu „rmdir". Es kann nur ein leeres Directory gelöscht werden.

Der Editor

Im folgenden soll der Editor beschrieben werden, der unter Unix am meisten verbreitet ist. Sein Name : „ed". Die wichtigsten Befehle werden soweit erläutert, daß der Anwender damit in der Lage ist, Texte einzugeben. Den gesamten Funktionsumfang von „ed" zu beschreiben, würde den Rahmen dieses Beitrags sprengen. „ed" ist ein zeilenorientierter Editor und außerdem extrem schweigsam. Da er keinen Prompt von sich gibt, sind risikofreudige Leute mit „ed" in ihrem Element. Die nahezu einzige Meldung, die man ihm entlocken kann ist ein „?" (z. B. bei Syntaxfehlern). „ed" kennt Kommandos, die aus einem Buchstaben bestehen und für sich alleine in einer Zeile stehen. Die Zeilen werden mit einer oder zwei Zeilennummern adressiert. Es gibt Befehle, die nur eine oder zwei Zeilennummern als Adressierung benötigen. Die Zeilennummern werden durch Komma getrennt, wenn mehr als eine erforderlich ist. Wird bei Befehlen, die nur eine Adresse benötigen, diese weggelassen, so wirken die Befehle auf die momentane Zeile. Hat man das Kommando zur Texteingabe eingetippt, so schaltet der Editor in den Eingabemodus um. Dieser wird durch eine Zeile, deren erstes und einziges Zeichen ein "." ist, beendet. Der Editor schreibt grundsätzlich nicht in die Originaldatei, sondern in einen Puffer. Erst wenn man den Schreibbefehl gibt, wird die Originaldatei überschrieben. In *Bild 11* wird eine typische Editorsitzung dargestellt. Die einzelnen Eingaben werden in der rechten Spalte erläutert.

`$ ed text` `?text`	`Aufruf von ed. Datei "text" soll editiert werden` `Datei "text" ist noch nicht vorhanden`

`a`
`Zeile 1`
`Zeile 2`
`Zeile 3`
`Zeile 4`
`.`
	`append: Anhängen an aktuelle Zeile`
	`Die Daten, die eingegeben werden`
	
	`Abschluß von append`

`1,$p`
`Zeile 1`
`Zeile 2`
`Zeile 3`
`Zeile 4`
	`print: Ausgabe der ersten bis letzten ($) Zeile`
	`Die Ausgabe von "p"`

`1,$n`
`1  Zeile 1`
`2  Zeile 2`
`3  Zeile 3`
`4  Zeile 4`
	`number: Ausgabe von erster bis letzter Zeile mit Numerierung`
	`Die Ausgabe von "n"`

`1,2m4`
`1,$p`
`Zeile 3`
`Zeile 4`
`Zeile 1`
`Zeile 2`
	`m: move. Bewege Zeile 1 bis 2 hinter Zeile 4.`

`1,2t4`
`1,$p`
`Zeile 3`
`Zeile 4`
`Zeile 1`
`Zeile 2`
`Zeile 3`
`Zeile 4`
	`t: transport. Kopiere Zeile 1...2 hinter Zeile 4`
	
	`Jetzt gibt es sechs Zeilen`

Bild 11. Der Unix-Editor „ed" weckt Erinnerungen an den MS-DOS-Editor Edlin

1,2d 1,$p Zeile 1 Zeile 2 Zeile 3 Zeile 4	d: delete. Lösche Zeile 1...2 Zwei Zeilen gelöscht. 4 Zeilen bleiben übrig
2m0 1,$n 1 Zeile 2 2 Zeile 1 3 Zeile 3 4 Zeile 4	"move" Zeile 2 hinter Zeile 0. Adressierung mit einer Adresse vorherige Zeile 2 ist neue Zeile 1. Vorherige Zeile 1 ist jetzt Zeile 2
2,3c das ist die neue Zeile und die dritte plus eine vierte, alles verschoben um 1 .	c: change. Ändere Zeile 2 und 3 "change" entspricht der Kombination aus "delete" und "append" Abschluß von "change"
1,$n 1 Zeile 2 2 das ist die neue zweite Zeile 3 und die dritte 4 plus eine vierte, alles verschoben um 1 5 Zeile 4	Zeile 2...4 ersetzen vorherige Zeilen 2 und 3
1i Zeile 1 Zeile 1a .	i: insert. Einschieben vor Zeile 1. Nur eine Zeilenadresse Die Daten für den Insert-Befehl Ende von "insert"
1,$n 1 Zeile 1 2 Zeile 1a 3 Zeile 2 4 das ist die neue zweite Zeile 5 und die dritte 6 plus eine vierte, alles verschoben um 1 7 Zeile 4	Zwei Zeilen vor der vorherigen ersten Zeile
4t5 1,$n 1 Zeile 1 2 Zeile 1a 3 Zeile 2 4 das ist die neue zweite Zeile 5 und die dritte 6 das ist die neue zweite Zeile 7 plus eine vierte, alles verschoben um 1 8 Zeile 4	Kopiere Zeile 4 hinter Zeile 5 Kopie der vierten Zeile
4s/ei/yy/p das ist die neue zwyyte Zeile	s:substitute. Ersetze in Zeile 4 /ei/ mit /yy/ und drucke Zeile nur erstes Vorkommen von ei ersetzt
4,6s/ei/yy/g 4,6p das ist die neue zwyyte Zyyle und die dritte das ist die neue zwyyte Zyyle	Ersetze in Zeile 4...6 /ei/ durch /yy/ Ausgabe Zeile 4...6 Alle /ei/ wurden mit /yy/ ersetzt (wegen g: global)
f text w 148 w text.dup 148 1,2w text,2 zeil 17 . das ist die neue zwyyte Zyyle a noch eine Aenderung .	Abfrage des Dateinamens Datei "text", wegen Aufruf "ed text" w: write. Schreibe Puffer nach Datei 148 Zeichen wurden geschrieben Schreibe Puffer nach Datei text.dup Schreibe Zeile 1 und 2 in Datei text.2zeil Ausgabe der momentanen Zeile Append nach momentaner Zeile Abschluß von append
1,$n 1 Zeile 1 2 Zeile 1a 3 Zeile 2 4 das ist die neue zwyyte Zyyle 5 und die dritte 6 das ist die neue zwyyte Zyyle 7 noch eine Aenderung 8 plus eine vierte, alles verschoben um 1 9 Zeile 4	Letzter Print-Befehl wirkt bis hier. Momentane Zeile Mit append angehängt
q ? w 168 q	q: quit. Beenden Letzte Änderung nicht geschrieben Schreiben nach Datei text Editorsitzung beenden

Holger Haase, Michael Menrad

mc-AllProg

PALs, EPROMs, PROMs, ... auf einem Gerät programmieren, Teil 2

Im letzten Monat wurde die Hardware des Programmiersystems mc-AllProg vorgestellt. Nun soll auf die Programmierverfahren eingegangen werden, die in der Software implementiert sind. Danach wird das Protokoll beschrieben, das für die Kommunikation zwischen AllProg und Computer verwendet wird. So können Sie die Hardware auch für eigene Treibersoftware nutzen.

Einleitend ein paar Worte, die für alle PALs gelten. Zunächst: der Name PAL (Programmable Array Logic) ist von MMI geschützt, weshalb Bausteine anderer Firmen oft PLAs (Programmable Logic Array) heißen.Wir werden hier durchgehend von PALs sprechen, ohne Rücksicht auf Schutzmarken. Den Hauptteil eines PALs stellt ein Feld dar, in dem man die Signale der Eingangspins – oder ihr Komplement – den Eingängen verschiedener UND-Gatter zuführen kann. Ein Ausschnitt aus einem solchen Feld ist in *Bild 1* links symbolisch dargestellt. Man erkennt vertikal verlaufende Leitungen (im Slang: Input Lines), auf denen die Eingangssignale und ihre Komplemente verfügbar sind, sowie eine horizontal verlaufende Bahn, die zu einem UND-Gatter führt. Diese Bahn mit UND-Gatter symbolisiert die rechts gezeigte Anordnung, in der das UND ebensoviele Eingänge besitzt, wie es vertikale Leitungen gibt und in der die vertikalen Leitungen an jeweils genau einen UND-Eingang über eine Durchschmelzverbindung angeschlossen sind. Diese Durchschmelzverbindungen können durch die Programmierung zerstört werden, so daß ein Eingang des UND-Gatters gezielt vom zugehörigen Eingangssignal oder von dessen Komplement abgekoppelt werden kann. Die interne Schaltung eines solchen UND-Gatters ist so ausgelegt, daß ein freier Eingang so wirkt, als ob er auf High-Pegel gelegt wäre. Ein solcher freier Eingang beeinflußt das UND-Gatter dann nicht mehr. Man kann also bei einem solchen UND-Gatter durch Programmieren frei bestimmen, auf welches Bitmuster (oder welche Bitmustergruppen) an den PAL-Eingängen hin es mit High-Pegel an seinem Ausgang antworten soll. Der Teil eines PALs, der den Ausgängen der UND-Gatter folgt, variiert von PAL-Typ zu PAL-Typ. Im allgemeinen werden die Ausgänge mehrerer UND-Gatter einem ODER-Gatter zugeführt, so daß man die Terme für die PAL-Ausgänge in der disjunktiven Normalform angeben kann. Der Praktiker spricht von einer Summe von Produkten.

20polige MMI-PALs

Die internen UND-Eingangsbündel (Produkt-Terme, im Slang auch Product Lines genannt) der MMI-PALs sind in zwei Blöcken zusammengefaßt. Den ersten Block bilden die Bündel 0 bis 31, den zweiten die Bündel 32 bis 63. Möchte man eine bestimmte Durchschmelzverbindung (neudeutsch: Fuse) ansprechen, so muß zunächst der Block festgelegt werden, in dem sie sich befindet. Liegt sie innerhalb des ersten Blocks, so erfolgt die Zuordnung der Signalnamen zu den Pins wie in *Bild 2a* dargestellt, sonst wie in *Bild 2b*. Die für die Spannungen benutzten Bezeichnungen und dazugehörigen Werte sind:

VL : Low-Pegel (0...0,8 Volt)
VH : High-Pegel (mindestens 2,0 Volt)
VHH: Programmierspannung (11,5...12 Volt)
VP : Security Fuse Programmierspannung (18,0...19,0 Volt)

Die Programmierung

Zum Start des Programmiervorgangs mit Auswahl des gewünschten Blocks (siehe oben) müssen der entsprechende OD-Pin auf VHH und der CLOCK-Pin auf Masse gelegt werden. Anschließend muß die zu der ausgewählten Durchschmelzverbindung gehörende Input Line selektiert werden, indem man den L/R-Pin und die Input-Pins mit den Pegeln nach *Bild 3* beschaltet. Die beiden Blökke sind in vier Gruppen zu je acht Produkttermen unterteilt. Die Lage des erwünschten Produktterms innerhalb einer solchen Gruppe wird dem PAL durch die Adreßpins A0 bis A2 mitgeteilt (siehe *Bild 4*). Nach dieser Anwahl gelingt der Schmelzprozeß, indem man die Versorgungsspannung (Vcc) von 5,0 Volt auf VHH erhöht und dann die Gruppe durch einen VHH-Puls von 20 µs Dauer auswählt (siehe auch Bild 4). Durch diesen Puls wird die nun vollständig adressierte Durchschmelzverbindung zerstört. Danach muß die Versorgungsspannung wieder auf 5,0 Volt abgesenkt werden und man kann die nächste Verbindung ins Visier nehmen. In *Bild 5* ist der Vorgang noch einmal als Zeitdiagramm dargestellt.

Das Auslesen

Zum Auslesen wird der OD-Pin auf VHH und der CLOCK-Pin auf VL gelegt. Wie oben beim Programmieren beschrieben, besorgen die Pins I0...I7, L/R und A0...A2 die weitere Adressierung – mit Ausnahme der Gruppenauswahl. Wird der CLOCK-Pin nach vollständiger Adressierung für mindestens 100 ns von VL auf VH gebracht, so kann man anschließend am zur gewünschten Durch-

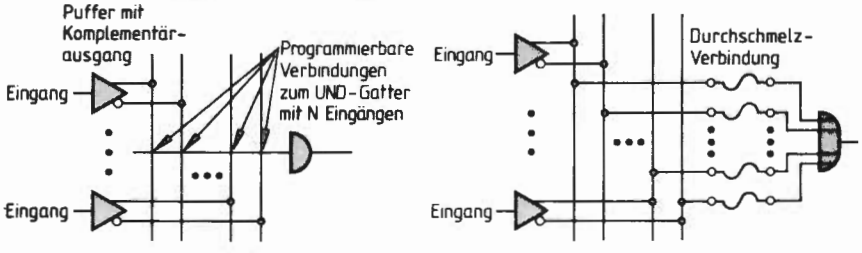


Bild 1. Das Innere von PALs

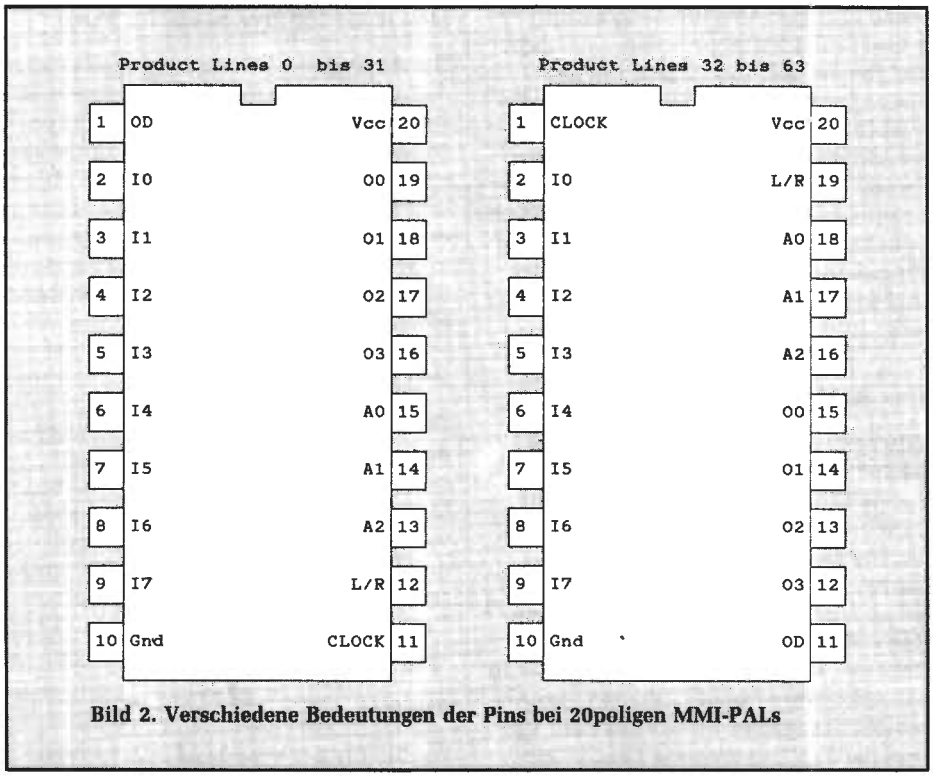


Bild 2. Verschiedene Bedeutungen der Pins bei 20poligen MMI-PALs

schmelzverbindung gehörenden O-Pin deren Zustand ablesen. Das Zeitdiagramm dieses Vorgangs zeigt *Bild 6*.

Das Brennen der „Security Fuse(s)"

Viele PALs kann man vor dem Auslesen schützen. Dazu gibt es bei den MMI-PALs zwei sogenannte „Security Fuses". Nachdem diese Durchschmelzverbindungen durchgebrannt sind, erhält man beim Auslesen des PALs völlig willkürliche Ergebnisse. Der interne Zustand eines so geschützten Bausteines ist nicht mehr zugänglich. Die beiden Security-Fuses werden folgendermaßen zerstört:
Zunächst wird Vcc auf 6,0 Volt gelegt (Bei PALs von Texas Instruments werden 0 Volt verlangt). Dann wird die erste Verbindung gebrannt, indem Pin 1 auf VP gelegt wird. Unter Umständen muß man Pin 1 auch mehrere Pulse zuführen (bis zu 5 sind zulässig). Die zweite Sicherung wird über Pin 13 gebrannt. Dadurch, daß diese PALs über getrennte Security Fuses für jeden Block verfügen, wäre es denkbar, nur einen der beiden Blöcke zu schützen.

24polige MMI-PALs

Im Gegensatz zu den 20poligen PALs von MMI sind die 24poligen in fünf Gruppen mit jeweils acht Produkttermen aufgeteilt. Die sich hieraus ergebenden Änderungen bezüglich des Programmierens und Auslesens sind *Bild 7, 8 und 9* zu entnehmen. Die Spannungswerte und Programmierzeiten entsprechen denen der 20poligen PALs.

20polige AMD-PALs

Im Gegensatz zu den PALs von MMI sind die PALs der Firma AMD nicht in zwei Blöcke unterteilt. Stattdessen sind diese PALs in acht Gruppen mit jeweils acht Produkttermen aufgeteilt. Deshalb ist die Funktion der Pins eindeutig. So adressieren die Pins 2...9 die Input Lines und die Lage der gewünschten Durchschmelzverbindung innerhalb einer Gruppe. Die Pins 12...19 werden dann zum endgültigen Anwählen und Brennen bzw. Lesen einer Verbindung benutzt. Zur Steuerung dienen die Pins 1 und 11. Die beiden übriggebliebenen Pins werden zum Anlegen der Versorgungsspannung benötigt, die beim Programmieren und Auslesen etwa 5,2 Volt betragen sollte. Wie bei den MMI-PALs, so auch hier wieder die für die Spannungen benutzten Bezeichnungen mit den dazugehörigen Werten:
VL : Low-Pegel (0...0,5 Volt)
VH : High-Pegel (2,4...5,5 Volt)
VHH: Spannung für die Kontroll-Pins (10...12 Volt)
VOP: Programmierspannung (18...22 Volt).

Input-Line-Nummer	Pin-Zustände								
	I7	I6	I5	I4	I3	I2	I1	I0	L/R
0	HH	HH	HH	HH	HH	HH	HH	L	Z
1	HH	HH	HH	HH	HH	HH	HH	H	Z
2	HH	HH	HH	HH	HH	HH	HH	L	HH
3	HH	HH	HH	HH	HH	HH	HH	H	HH
4	HH	HH	HH	HH	HH	HH	L	HH	Z
5	HH	HH	HH	HH	HH	HH	H	HH	Z
6	HH	HH	HH	HH	HH	HH	L	HH	HH
7	HH	HH	HH	HH	HH	HH	H	HH	HH
8	HH	HH	HH	HH	HH	L	HH	HH	Z
9	HH	HH	HH	HH	HH	H	HH	HH	Z
10	HH	HH	HH	HH	HH	L	HH	HH	HH
11	HH	HH	HH	HH	HH	H	HH	HH	HH
12	HH	HH	HH	HH	L	HH	HH	HH	Z
13	HH	HH	HH	HH	H	HH	HH	HH	Z
14	HH	HH	HH	HH	L	HH	HH	HH	HH
15	HH	HH	HH	HH	H	HH	HH	HH	HH
16	HH	HH	HH	L	HH	HH	HH	HH	Z
17	HH	HH	HH	H	HH	HH	HH	HH	Z
18	HH	HH	HH	L	HH	HH	HH	HH	HH
19	HH	HH	HH	H	HH	HH	HH	HH	HH
20	HH	HH	L	HH	HH	HH	HH	HH	Z
21	HH	HH	H	HH	HH	HH	HH	HH	Z
22	HH	HH	L	HH	HH	HH	HH	HH	HH
23	HH	HH	H	HH	HH	HH	HH	HH	HH
24	HH	L	HH	HH	HH	HH	HH	HH	Z
25	HH	H	HH	HH	HH	HH	HH	HH	Z
26	HH	L	HH	HH	HH	HH	HH	HH	HH
27	HH	H	HH	HH	HH	HH	HH	HH	HH
28	L	HH	HH	HH	HH	HH	HH	HH	Z
29	H	HH	HH	HH	HH	HH	HH	HH	Z
30	L	HH	HH	HH	HH	HH	HH	HH	HH
31	H	HH	HH	HH	HH	HH	HH	HH	HH

L = VL H = VH
HH = VHH Z = Tristate

Bild 3. Schema zum Programmieren der 20poligen MMI-PALs

Product-Line-Nummer	Pin-Zustände						
	O3	O2	O1	O0	A2	A1	A0
0, 32	Z	Z	Z	HH	Z	Z	Z
1, 33	Z	Z	Z	HH	Z	Z	HH
2, 34	Z	Z	Z	HH	Z	HH	Z
3, 35	Z	Z	Z	HH	Z	HH	HH
4, 36	Z	Z	Z	HH	HH	Z	Z
5, 37	Z	Z	Z	HH	HH	Z	HH
6, 38	Z	Z	Z	HH	HH	HH	Z
7, 39	Z	Z	Z	HH	HH	HH	HH
8, 40	Z	Z	HH	Z	Z	Z	Z
9, 41	Z	Z	HH	Z	Z	Z	HH
10, 42	Z	Z	HH	Z	Z	HH	Z
11, 43	Z	Z	HH	Z	Z	HH	HH
12, 44	Z	Z	HH	Z	HH	Z	Z
13, 45	Z	Z	HH	Z	HH	Z	HH
14, 46	Z	Z	HH	Z	HH	HH	Z
15, 47	Z	Z	HH	Z	HH	HH	HH
16, 48	Z	HH	Z	Z	Z	Z	Z
17, 49	Z	HH	Z	Z	Z	Z	HH
18, 50	Z	HH	Z	Z	Z	HH	Z
19, 51	Z	HH	Z	Z	Z	HH	HH
20, 52	Z	HH	Z	Z	HH	Z	Z
21, 53	Z	HH	Z	Z	HH	Z	HH
22, 54	Z	HH	Z	Z	HH	HH	Z
23, 55	Z	HH	Z	Z	HH	HH	HH
24, 56	HH	Z	Z	Z	Z	Z	Z
25, 57	HH	Z	Z	Z	Z	Z	HH
26, 58	HH	Z	Z	Z	Z	HH	Z
27, 59	HH	Z	Z	Z	Z	HH	HH
28, 60	HH	Z	Z	Z	HH	Z	Z
29, 61	HH	Z	Z	Z	HH	Z	HH
30, 62	HH	Z	Z	Z	HH	HH	Z
31, 63	HH	Z	Z	Z	HH	HH	HH

HH = VHH Z = Tristate

Bild 4. Zur Adressierung der Produkt-Term-Gruppen

Wie man sieht, weichen die Programmierspannungen ziemlich stark von den bei MMI-PALs verlangten ab. Ein weitaus größerer Unterschied liegt aber in der beim Schalten erforderlichen Flankensteilheit. Werden bei MMI-PALs nur recht gemächliche 1 bis 10 Volt/µs verlangt, so fordern die PALs der Firma AMD immerhin 100 bis 250 Volt/µs, wenn man sie mit Erfolg programmieren will.

Über die Pins 5...9 (siehe *Bild 10*) sowie die Pins 2...4 (*Bild 11*) werden acht Durchschmelzverbindungen ausgewählt, auf die man anschließend über die Pins 12...19 zugreifen kann. Dann wird Pin 1 auf VHH gelegt, um dem Baustein mitzuteilen, daß man ihn programmieren möchte. Kurz danach wird der Verbindung, die man durchbrennen möchte, über den entsprechenden Anschluß (siehe ebenfalls Bild 11) die Programmierspannung VOP zugeführt. Der eigentliche Brennvorgang beginnt dann, wenn die Spannung an Pin 11 von VL auf VHH erhöht wird. Ist die Programmierzeit abgelaufen, wird der Pin, der die Programmierspannung zur Verbindung führte, wieder in den Tristate-Zustand versetzt. Anschließend kehren Pin 1 und Pin 11 zu ihren Anfangsspannungen (bei beiden VL) zurück, um den Brennprozeß zu beenden. Nun wird die Verbindung, wie im folgenden Abschnitt dargestellt, ausgelesen und mit ihrem Soll-Zustand verglichen. War der Brennvorgang nicht erfolgreich, so wird er mit einer um den Faktor 100 längeren Programmierzeit (aus 50 µs werden 5 ms) wiederholt. Den zeitlichen Ablauf zeigt *Bild 12*.

Das Auslesen

Wie beim Programmieren werden zunächst einmal durch die Adressierpins 2...9 acht Verbindungen ausgewählt. Pin 11 wird anschließend auf VHH geschaltet. Nachdem Pin 1 (war bis jetzt die ganze Zeit auf VL) für kurze Zeit auf VH gelegt wurde, um bei Register-PALs die Register zu takten, kann man an den Pins 12...19 den Zustand der acht Durchschmelzverbindungen abfragen. Das Absenken der Spannung an Pin 11 auf VL beendet den Lesevorgang. Den Ablauf des Lesevorgangs soll *Bild 13* verdeutlichen.

Das Brennen der Security Fuse

Bei AMD-PALs hat man es nur mit einer Security Fuse zu tun. Sie wird gebrannt, indem Pin 11 ein 50 µs langer Puls von Masse zu VOP zugeführt wird, nachdem zuvor die Spannung an Pin 5 auf VHH erhöht wurde. Wenn die Security Fuse erfolgreich gebrannt wurde, so müssen alle programmierbaren Verbindungen des Bausteins beim Lesen als gebrannt erscheinen.

PALs anderer Firmen

Die PALs der Firma National Semiconductor verhalten sich beim Auslesen und Programmieren genauso wie die MMI-PALs. Das gleiche gilt, abgesehen von der Programmierspannung VHH und der Spannung VP zum Programmieren der Security Fuse, für die PALs von Texas Instruments. Bei diesen muß VHH 10,5 Volt und VP 21,0 Volt betragen.

PROMs

Da die von den verschiedenen PROMs geforderten Programmierverfahren zu vielfältig sind, als daß man sie im Rah-

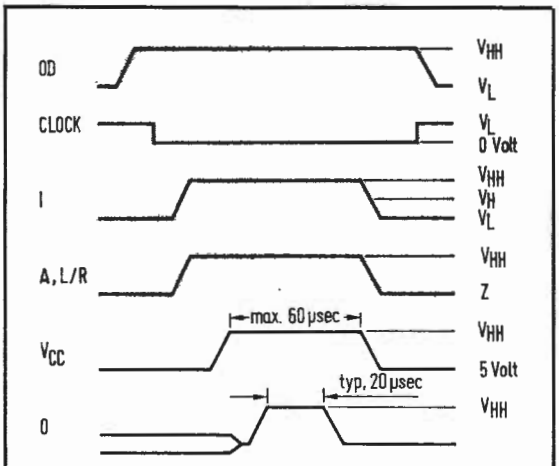


Bild 5. Das Timing beim Programmieren

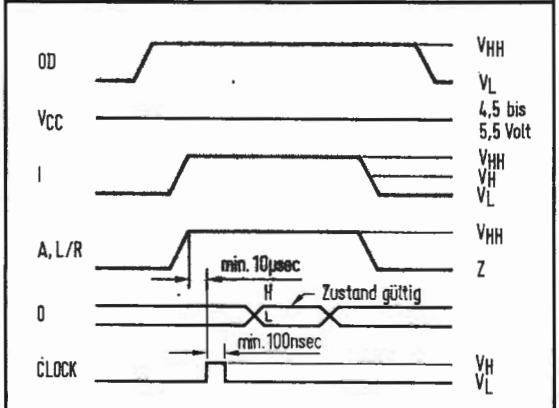


Bild 6. Das Timing beim Lesen

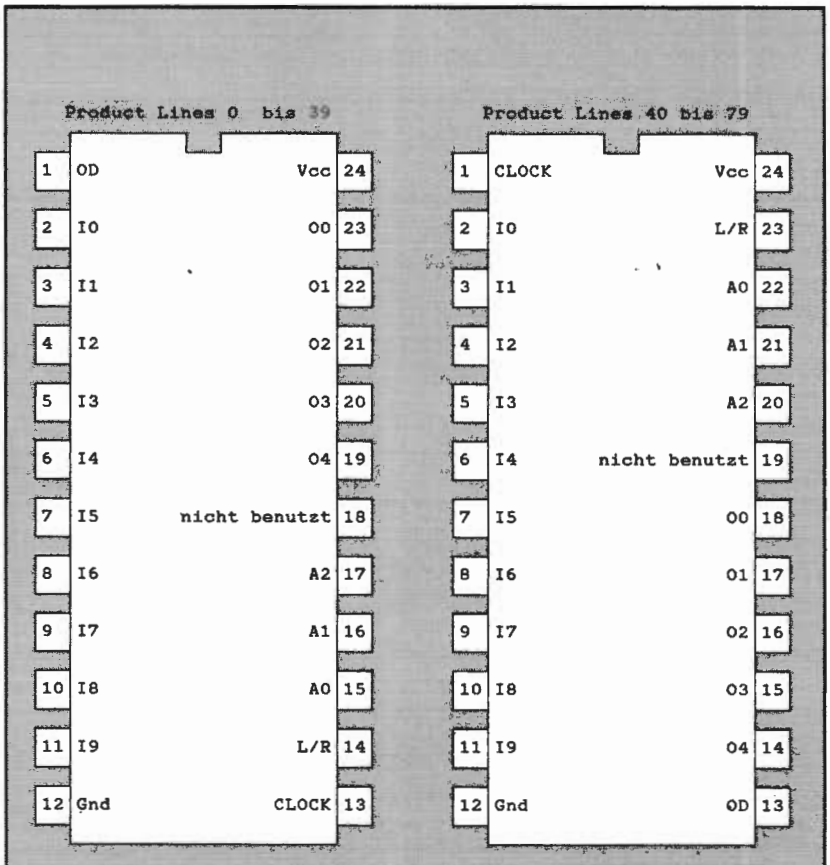


Bild 7. Die Pinbelegungen der 24poligen MMI-PALs

men dieses Artikels vollständig beschreiben könnte, soll im folgenden nur auf ein Verfahren näher eingegangen werden. Es läßt sich jedoch sagen, daß die Verfahren zum Programmieren von PROMs stark dem Programmierverfahren für AMD-PALs ähneln. So wird auch hier zuerst eine Speicherstelle adressiert und dann Bit für Bit deren Inhalt programmiert. Hier nun zu dem Verfahren, mit dem sämtliche zur Zeit in AllProg implementierten PROMs programmieren lassen. Zuerst wird durch die Adreßpins des Bausteins die Speicherstelle bestimmt, die behandelt werden soll. Dies geschieht mit ganz normalen TTL-Pegeln. Anschließend wird das PROM in den Programmier-Modus versetzt, indem die Spannung am Vcc-Pin auf 8,75 Volt erhöht wird. Nach einer gewissen Verzögerungszeit wird dann dem Output-Pin, der zu dem Bit gehört, das gebrannt (auf 1 gesetzt) werden soll, eine Spannung von 17,5 Volt zugeführt. Die eigentliche Zerstörung der internen Verbindung, die das Bit auf 0 legt, erfolgt dadurch, daß der CE-Pin (Chip Enable) für 10 bis 25 Mikrosekunden auf TTL-Low-Pegel gelegt wird. Nun wird die Spannung am Output-Pin wieder gesenkt, und man kann sich dem nächsten Bit zuwenden. Hierbei gilt es zu beachten, daß nur solche Bits gebrannt werden sollten, bei denen eine logische 1 erwünscht ist.

Das Programmieren von EPROMs

Die hier verwendeten Programmierverfahren „Normal", „Interaktiv" und „Intelligent" wurden bereits in mc 1987, Ausgabe 1, beschrieben. Deshalb soll hier nur auf den neu dazugekommenen Programmiermodus „Quick" eingegangen werden. Dabei handelt es sich um ein Programmierverfahren, das mit extrem kurzen Programmierpulsen auskommt. Es ist wegen der kurzen Programmierzeiten besonders geeignet, um große EPROMs zu programmieren (*Bild 14*). Leider ist seine Anwendbarkeit auch auf diese großen Speicher begrenzt.

Das Protokoll der Schnittstelle

Würde die Kommunikation während des Programmierens bei großen EPROMs zum Beispiel im häufig verwendeten Intel-Hex-Format absolviert werden, würde das sehr lange dauern. Man müßte pro acht Bit von oder zum Baustein mehr als zwei Byte übertragen. Andererseits müssen Datensicherheit und Flexibilität bei der Kommunikation gewahrt bleiben. Es sollen immerhin verschiedene Bausteinfamilien mit den unterschiedlichsten integrierten Schaltungen bearbeitbar sein. Die Kommunikation sollte weiterhin auch an Rechnern laufen, über deren Betriebssystem (z.B. CP/M-80) der Status der seriellen Verbindung nicht abfragbar ist. Dies alles hat zu einem neuen Protokoll geführt.
Die Hardware-Verbindung zur Schaltung ist 8 Bit breit. Es wird im folgenden nun zwischen einzelnen Bytes und Datenblöcken unterschieden. Als einzelne Bytes gibt es Anforderungs-, Quittungs-, Fehler- und Time-Out-Signale. Die

Input-Line-Nummer	Pin-Zustände										
	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0	L/R
0	HH	HH	HH	HH	HH	HH	HH	HH	HH	L	Z
1	HH	HH	HH	HH	HH	HH	HH	HH	HH	H	Z
2	HH	HH	HH	HH	HH	HH	HH	HH	HH	L	HH
3	HH	HH	HH	HH	HH	HH	HH	HH	HH	H	HH
4	HH	HH	HH	HH	HH	HH	HH	HH	L	HH	Z
5	HH	HH	HH	HH	HH	HH	HH	HH	H	HH	Z
6	HH	HH	HH	HH	HH	HH	HH	HH	L	HH	HH
7	HH	HH	HH	HH	HH	HH	HH	HH	H	HH	HH
8	HH	HH	HH	HH	HH	HH	HH	L	HH	HH	Z
9	HH	HH	HH	HH	HH	HH	HH	H	HH	HH	Z
10	HH	HH	HH	HH	HH	HH	HH	L	HH	HH	HH
11	HH	HH	HH	HH	HH	HH	HH	H	HH	HH	HH
12	HH	HH	HH	HH	HH	HH	L	HH	HH	HH	Z
13	HH	HH	HH	HH	HH	HH	H	HH	HH	HH	Z
14	HH	HH	HH	HH	HH	HH	L	HH	HH	HH	HH
15	HH	HH	HH	HH	HH	HH	H	HH	HH	HH	HH
16	HH	HH	HH	HH	HH	L	HH	HH	HH	HH	Z
17	HH	HH	HH	HH	HH	H	HH	HH	HH	HH	Z
18	HH	HH	HH	HH	HH	L	HH	HH	HH	HH	HH
19	HH	HH	HH	HH	HH	H	HH	HH	HH	HH	HH
20	HH	HH	HH	HH	L	HH	HH	HH	HH	HH	Z
21	HH	HH	HH	HH	H	HH	HH	HH	HH	HH	Z
22	HH	HH	HH	HH	L	HH	HH	HH	HH	HH	HH
23	HH	HH	HH	HH	H	HH	HH	HH	HH	HH	HH
24	HH	HH	HH	L	HH	HH	HH	HH	HH	HH	Z
25	HH	HH	HH	H	HH	HH	HH	HH	HH	HH	Z
26	HH	HH	HH	L	HH	HH	HH	HH	HH	HH	HH
27	HH	HH	HH	H	HH	HH	HH	HH	HH	HH	HH
28	HH	HH	L	HH	HH	HH	HH	HH	HH	HH	Z
29	HH	HH	H	HH	HH	HH	HH	HH	HH	HH	Z
30	HH	HH	L	HH	HH	HH	HH	HH	HH	HH	HH
31	HH	HH	H	HH	HH	HH	HH	HH	HH	HH	HH
32	HH	L	HH	HH	HH	HH	HH	HH	HH	HH	Z
33	HH	H	HH	HH	HH	HH	HH	HH	HH	HH	Z
34	HH	L	HH	HH	HH	HH	HH	HH	HH	HH	HH
35	HH	H	HH	HH	HH	HH	HH	HH	HH	HH	HH
36	L	HH	HH	HH	HH	HH	HH	HH	HH	HH	Z
37	H	HH	HH	HH	HH	HH	HH	HH	HH	HH	Z
38	L	HH	HH	HH	HH	HH	HH	HH	HH	HH	HH
39	H	HH	HH	HH	HH	HH	HH	HH	HH	HH	HH

L = VL H = VH
HH = VHH Z = Tristate

Bild 8. Programmier-Pegel-Schema für 24polige MMI-PALs

Product-Line-Nummer	Pin-Zustände							
	O4	O3	O2	O1	O0	A2	A1	A0
0, 40	Z	Z	Z	Z	HH	Z	Z	Z
1, 41	Z	Z	Z	Z	HH	Z	Z	HH
2, 42	Z	Z	Z	Z	HH	Z	HH	Z
3, 43	Z	Z	Z	Z	HH	Z	HH	HH
4, 44	Z	Z	Z	Z	HH	HH	Z	Z
5, 45	Z	Z	Z	Z	HH	HH	Z	HH
6, 46	Z	Z	Z	Z	HH	HH	HH	Z
7, 47	Z	Z	Z	Z	HH	HH	HH	HH
8, 48	Z	Z	Z	HH	Z	Z	Z	Z
9, 49	Z	Z	Z	HH	Z	Z	Z	HH
10, 50	Z	Z	Z	HH	Z	Z	HH	Z
11, 51	Z	Z	Z	HH	Z	Z	HH	HH
12, 52	Z	Z	Z	HH	Z	HH	Z	Z
13, 53	Z	Z	Z	HH	Z	HH	Z	HH
14, 54	Z	Z	Z	HH	Z	HH	HH	Z
15, 55	Z	Z	Z	HH	Z	HH	HH	HH
16, 56	Z	Z	HH	Z	Z	Z	Z	Z
17, 57	Z	Z	HH	Z	Z	Z	Z	HH
18, 58	Z	Z	HH	Z	Z	Z	HH	Z
19, 59	Z	Z	HH	Z	Z	Z	HH	HH
20, 60	Z	Z	HH	Z	Z	HH	Z	Z
21, 61	Z	Z	HH	Z	Z	HH	Z	HH
22, 62	Z	Z	HH	Z	Z	HH	HH	Z
23, 63	Z	Z	HH	Z	Z	HH	HH	HH
24, 64	Z	HH	Z	Z	Z	Z	Z	Z
25, 65	Z	HH	Z	Z	Z	Z	Z	HH
26, 66	Z	HH	Z	Z	Z	Z	HH	Z
27, 67	Z	HH	Z	Z	Z	Z	HH	HH
28, 68	Z	HH	Z	Z	Z	HH	Z	Z
29, 69	Z	HH	Z	Z	Z	HH	Z	HH
30, 70	Z	HH	Z	Z	Z	HH	HH	Z
31, 71	Z	HH	Z	Z	Z	HH	HH	HH
32, 72	HH	Z	Z	Z	Z	Z	Z	Z
33, 73	HH	Z	Z	Z	Z	Z	Z	HH
34, 74	HH	Z	Z	Z	Z	Z	HH	Z
35, 75	HH	Z	Z	Z	Z	Z	HH	HH
36, 76	HH	Z	Z	Z	Z	HH	Z	Z
37, 77	HH	Z	Z	Z	Z	HH	Z	HH
38, 78	HH	Z	Z	Z	Z	HH	HH	Z
39, 79	HH	Z	Z	Z	Z	HH	HH	HH

HH = VHH Z = Tristate

Bild 9. Adressier-Schema für 24polige MMI-PALs

Input-Line-Nummer	Zustände der Adreßpins				
	Pin9	Pin8	Pin7	Pin6	Pin5
0	L	L	L	L	L
1	L	L	L	L	H
2	L	L	L	H	L
3	L	L	L	H	H
4	L	L	H	L	L
5	L	L	H	L	H
6	L	L	H	H	L
7	L	L	H	H	H
8	L	H	L	L	L
9	L	H	L	L	H
10	L	H	L	H	L
11	L	H	L	H	H
12	L	H	H	L	L
13	L	H	H	L	H
14	L	H	H	H	L
15	L	H	H	H	H
16	H	L	L	L	L
17	H	L	L	L	H
18	H	L	L	H	L
19	H	L	L	H	H
20	H	L	H	L	L
21	H	L	H	L	H
22	H	L	H	H	L
23	H	L	H	H	H
24	H	H	L	L	L
25	H	H	L	L	H
26	H	H	L	H	L
27	H	H	L	H	H
28	H	H	H	L	L
29	H	H	H	L	H
30	H	H	H	H	L
31	H	H	H	H	H

L = VL H = VH

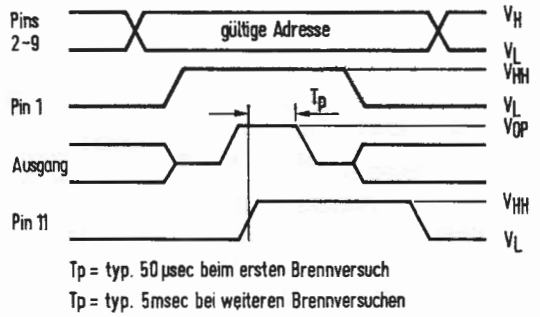
Bild 10. Die Anwahl der Input Lines...

Product-Line-Nummer								Zustände der Adreßpins Pin4	Pin3	Pin2
0	8	16	24	32	40	48	56	L	L	L
1	9	17	25	33	41	49	57	L	L	H
2	10	18	26	34	42	50	58	L	H	L
3	11	19	27	35	43	51	59	L	H	H
4	12	20	28	36	44	52	60	H	L	L
5	13	21	29	37	45	53	61	H	L	H
6	14	22	30	38	46	54	62	H	H	L
7	15	23	31	39	47	55	63	H	H	H
Pin 19	Pin 18	Pin 17	Pin 16	Pin 15	Pin 14	Pin 13	Pin 12			

Programmier- und Verify-Pin

L = VL
H = VH

Bild 11. ... und der Product Lines



Tp = typ. 50 µsec beim ersten Brennversuch
Tp = typ. 5msec bei weiteren Brennversuchen

Bild 12. Das Programmier-Timing für die 20poligen AMD-PALs

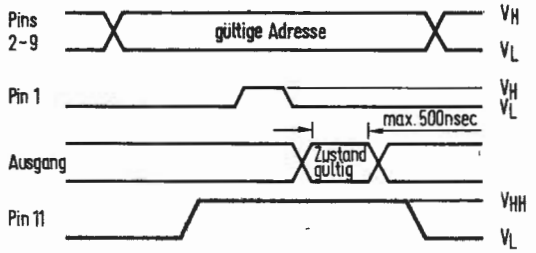


Bild 13. Das Lese-Timing dazu

Time-Out-Bytes, welche Leersignale darstellen, werden alle 0,5 Sekunden gesendet. Man kann sie zum Befreien aus Endlos-Lese-Warteschleifen verwenden. Die verschiedenen Anforderungs-Bytes und ihre Bedeutung zeigt *Bild 15*. Die Quittungs-Bytes, die AllProg sendet, sind in *Bild 16* aufgeführt. Das Programmiergerät sendet nie Daten oder Fehlerblöcke, ohne ausdrücklich dazu aufgefordert worden zu sein. Von ihm sind also normalerweise einzelne Bytes zu erwarten. Die Blöcke (*Bild 17*) haben dagegen folgenden Aufbau: Ihr erstes Byte ist eine 90. Dieses Synchronisationszeichen kennzeichnet den Blockanfang. Es folgt ein Byte, dessen Bit 7 die Art des Blocks definiert. Die restlichen Bits dienen zur Übermittlung der Blocklänge (in Bytes). Unter der Art des Blocks wird beim Senden zu AllProg zwischen Datenblock (Bit 7=0) und Befehlsblock (Bit 7=1) unterschieden. Bei Sendungen von AllProg steht dieses Bit für die Kennzeichnung eines Fehlerblocks (Bit 7=1). Dann werden die angemeldete Anzahl von Bytes übermittelt. Zum Schluß gibt es ein Prüfsummen-Byte, in dem die Summe aller vorangegangenen Zeichen (inkl. des Startzeichens 90) steht. *Bild 18* zeigt die Block-Typen im Überblick. Handelt es sich bei einem vom Computer gesendeten Block um einen Befehlsblock, so haben seine Bytes (zwischen Längen- und Prüfsummen-Byte) eine festgelegte Bedeutung (*Bild 19*). In dem Befehl wird alles weitere für die dann folgende Behandlung eines Bausteins festgelegt. Um nach dem Senden des Befehls einen Baustein einlegen zu können, muß das Bit 7 des Bytes „Art der Operation" gesetzt sein. Damit schaltet man alle Pins des Programmiersockels hochohmig, so daß ein Baustein beim Einlegen nicht beschädigt wird. Zur Kontrolle erlischt die LED neben dem Sockel. Auch die Fehlermeldungen, die von der Programmierkarte zurückgesen-

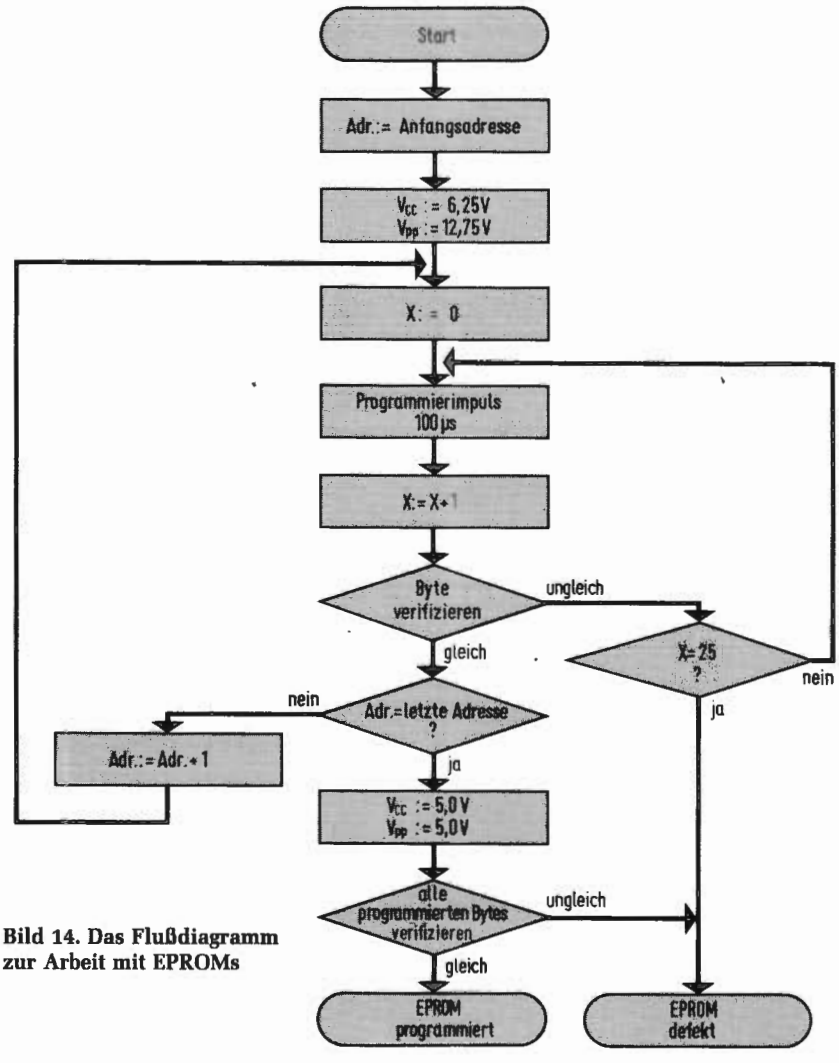


Bild 14. Das Flußdiagramm zur Arbeit mit EPROMs

det werden, haben einen festen Aufbau (*Bild 20*). Die ersten beiden Bytes sind dabei am wichtigsten. Im ersten Byte steht eine Nummer (HARD-Fehler) größer Null, wenn an der Hardware der Programmierkarte eine Betriebsstörung festgestellt wurde. Im zweiten Byte (PROG-Fehler) werden Probleme mit dem behandelten Baustein gemeldet. Läßt sich zum Beispiel ein EPROM nicht programmieren, weil es nicht gelöscht wurde, so stünde in PROG-Fehler eine „1" für „nicht leer". In „gelesenem Byte" und „Adresse" würde das erste nicht gelöschte Byte spezifiziert werden. Die wichtigsten Fehler zeigt *Bild 21*. Zum Schluß des Fehlerblocks werden einige Analogwerte übermittelt. Dabei besitzt der Wert für die Temperatur keine genormte Einheit. Sollte er unter 60 sinken, so wird es dem Gerät (genauer: dem Kühlblech) zu warm! Sind Nummern innerhalb solcher Blöcke mehrere Bytes lang, so wird zuerst das niederwertigste und zum Schluß das höchstwertige Byte übertragen. Bevor nun auf die Besonderheiten bei einzelnen Bausteinfamilien eingegangen wird, sollen erst einmal zwei Beispiele für die Verwendung dieses Protokolls gegeben werden.

Lesen eines EPROMs

>Befehl
Im Befehlsblock an AllProg ist der Typ des Bausteins (2 = EPROM), die Typ-Nummer des ICs (z.B. 7 = 2732A) und die Art der Operation (1 = Lesen, Bit 7 gesetzt), sowie die Adresse des ersten zu lesenden Bytes festgelegt.

<Quittung
Ist dieses Byte Null, so kann der Befehl ausgeführt werden. Die LED ist dunkel. Nun wird der Baustein im Sockel befestigt. Sollte dieses Byte allerdings Eins sein, so ist ein Fehler aufgetreten. Der Befehl kann nicht ausgeführt werden. Um Näheres zu erfahren, kann man einen Fehlerblock anfordern.

>Anforderung
Jetzt werden mit einer Null Daten-Bytes von AllProg angefordert.

<Daten
AllProg sendet einen Block mit 16 Daten-Bytes oder einen Fehlerblock, falls das Lesen nicht möglich ist. Nach jedem gelesenen Byte wird die im Befehl spezifizierte Adresse um eins erhöht. So wird beim Lesevorgang kein weiterer Befehl benötigt.
Diese letzten beiden Punkte werden so oft wie nötig wiederholt.

>Ende
Mit diesem Byte wird AllProg das Ende der Operation mitgeteilt.

Programmieren eines EPROMs

>Befehl
Im Befehlsblock an AllProg ist der Typ des Bausteins (2 = EPROM), die Nummer des ICs (z.B. 7 = 2732A) und die Art der Operation (2 oder 3 = Programmieren, Bit 7 gesetzt), sowie die Adresse des ersten zu programmierenden Bytes festgelegt.

<Quittung
Ist dieses Byte Null, so kann der Befehl ausgeführt werden. Die LED ist dunkel.

Bild 15. Allprog sendet nur auf Anforderung

Byte	Bedeutung
0	Daten- bzw. Fehlerblock senden!
1	Fehlerblock senden!
2	Programmierplatine in Wartestellung!

Byte	Bedeutung
0	Befehl wurde empfangen und ausgeführt, kein Fehler aufgetreten.
1	Es ist ein Fehler aufgetreten.
240	Time Out

Bild 16. Das quittiert AllProg

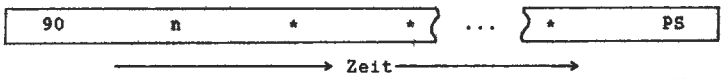


Bild 9. Der Aufbau eines Blocks

Bild 17. Ein AllProg-Block

BEFEHLE	mit ihnen werden dem Programmiergerät Baustein, Startadresse und ggf. die Nummer des Programmieralgorithmus mitgeteilt. Dabei handelt es sich auch um die Information, ob aus dem zu behandelnden IC gelesen oder in es programmiert werden soll. Diese Befehle kommen immer vom Computer, nie von der Programmierhardware.
DATEN	Sie enthalten Informationen, die aus einem Baustein gelesen wurden oder die in ihn programmiert werden sollen. Sie können demnach vom Computer oder von der Programmierkarte kommen.
FEHLER	Fehlermeldungen kommen nur von der Karte zum Computer.

Bild 18. Überblick über die Block-Arten

Byte	Bedeutung
0-1	Typ des Bausteins (1=PROM, 2=EPROM, 3=PAL)
2-3	Nummer des Bausteins
4	Art der Operation (z.B. Lesen, Programmieren, Leertest)
5-7	Startadresse (für folgende Datenblöcke)

Bild 19. So ist der 7-Byte-Befehl aufgebaut

Byte	Bedeutung
1	HARD-Fehler
2	PROG-Fehler
3	Software-Version
4	gelesenes Byte
5	Bausteintyp (s. Tabelle 10)
6-7	Nummer des Bausteins
8	Art der Operation
9-11	Adresse
12	Temperatur
13	Uext. (in 0,2 Volt)
14	Uccp (in 0,1 Volt)
15	Up1 (in 0,1 Volt)
16	Up2 (in 0,1 Volt)

Bild 20. Die Struktur des Fehler-Blockes

HARD-Fehler

3,19	Uext zu klein für Uccp
34	Uext zu klein für Up1
50	Uext zu klein für Up2
250	Temperatur zu hoch
252	Undefinierter Befehl
254	Blocklänge zu groß
255	Prüfsummenfehler

PROG-Fehler

1	Baustein nicht leer
2	Baustein läßt sich nicht programmieren

Bild 21. Die wichtigsten Fehlernummern bei Hardware und Programmierung

Byte	Bedeutung
1	--
2	0=leer, 1=nicht leer
3	Software-Version
4	gelesenes Byte
5	Bausteintyp
6-7	Nummer des Bausteins
8	Art der Operation
9-11	Adresse

Bild 22. Das sind die Mitteilungen des Leertestes

Nun wird der Baustein im Sockel befestigt. Sollte dieses Byte Eins sein, so ist ein Fehler aufgetreten. Der Befehl kann nicht ausgeführt werden. Um Näheres zu erfahren, kann man einen Fehlerblock anfordern. Auf keinen Fall dürfen dann Daten an AllProg gesendet werden!

>Daten
Nun erwartet AllProg einen Block mit maximal 16 Daten-Bytes. Nach jedem programmierten Byte wird die im Befehl spezifizierte Adresse um eins erhöht. So wird beim Programmiervorgang kein weiterer Befehl benötigt.

<Quittung
AllProg teilt mit einer Null eine erfolgreiche oder mit einer Eins eine fehlerhafte Programmierung mit. Es kann wieder ein Fehlerblock angefordert werden.

Diese letzten beiden Punkte werden so oft wie nötig wiederholt.

>Ende
Mit diesem Byte wird AllProg das Ende der Operation mitgeteilt.

Leertest

Für EPROMs und PROMs gibt es einen Befehl für einen kompletten Leertest. Das ergibt zwei Vorteile: AllProg muß nur einen Datenblock mit der Information voll/leer liefern. Es entsteht also kaum Zeitverlust durch die serielle Übertragung. Außerdem kennt AllProg die „Leerzustände" der Bausteine. So müssen in den Bits von gelöschten EPROMs Einsen stehen, dagegen steht bei einigen PROMs eine Null in einem nicht programmierten Bit. Der Ablauf eines solchen Leertests ist ähnlich dem eines Lesevorgangs. Als Startadresse muß 000000h vorgegeben werden. Nach der Aufforderung sendet AllProg dann einen 11-Byte-Datenblock, dessen Aufbau in *Bild 22* aufgeführt ist. Das wichtigste Byte ist das zweite. Ist es Null, so ist der Baustein leer. In anderen Fällen steht unter „gelesenes Byte" und „Adresse" die bereits programmierte Information. Für PALs gibt es diesen Leertest nicht. Sie müssen konventionell ausgelesen werden.

Datenblöcke

Ein Byte in einem Datenblock entspricht der Information unter einer Adresse in dem bearbeiteten Baustein. Handelt es sich dabei zum Beispiel um ein PROM mit nur 4 Bit Wortbreite, so steht die Information jeweils in den niederwertigen 4 Bit eines Datenblockbytes. Bei PALs wird in einem Byte des Datenblocks immer nur das Bit 0 verwendet. Es entspricht einer „Fuse". Eine Null steht für ein gebranntes bzw. für ein zu brennendes Bit („Fuse"). Die übrigen 7 Bit der Daten-Bytes müssen Null sein.

Adressen bei PALs

Eine Besonderheit bei PALs sind die Adreß-Bytes des Befehls. Das niederwertige Byte steht für die „Input Line". Die höherwertigen zwei Bytes enthalten die „Product Line". Da es keine 256 Input Lines gibt, muß nach jeder dieser Zeilen erneut ein Befehl gesendet werden, um die höherwertigen Adress-Bytes zu verändern. Soll die Auslesesicherung gebrannt werden, so sendet man einfach den Befehl „Programmieren" mit der Startadresse FFFFFFh. Danach folgen jedoch keine Datenblöcke, sondern es

Bausteintyp	Art der Operation	Bedeutung
1	0	PROM-Leertest
1	1	PROM-Lesen
1	2	PROM-Programmieren
2	0	EPROM-Leertest
2	1	EPROM-Lesen
2	2	EPROM-Programmieren (I)
2	3	EPROM-Programmieren (II)
3	0	PAL-Lesen bei Vcc= 4,5 Volt
3	1	PAL-Lesen bei Vcc= 5,0 Volt
3	2	PAL-Lesen bei Vcc= 5,5 Volt
3	3	PAL-Programmieren

Bild 23. Die möglichen Parameter-Kombinationen bei Befehlen

(E)EPROMs

Bausteinnummer	Baustein	Organisation (in KByte)	Programmiermodi (I)	Programmiermodi (II)
0	AllProg	16	-	-
1	2516 (25V)	2	Normal	Normal
2	2716 (25V)	2	Normal	Normal
3	48016 (25V)	2	Normal	-
4	2816 (21V)	2	Normal	-
5	2532 (25V)	4	Normal	Normal
6	2732 (25V)	4	Normal	Normal
7	2732A (21V)	4	Normal	Normal
8	2764 (21V)	8	Normal	Interaktiv
9	2764 (12,5V)	8	Normal	Interaktiv
10	27128 (21V)	16	Normal	Interaktiv
11	27128 (12,5V)	16	Normal	Interaktiv
12	27256 (21V)	32	Interaktiv	Intelligent
13	27256 (12,5V)	32	Interaktiv	Intelligent
14	27512 (21V)	64	Interaktiv	Intelligent
15	27512 (12,5V)	64	Interaktiv	Intelligent
16	27513 (12,5V)	64	Intelligent	Quick
16	27010 (12,5V)	128	Intelligent	Quick
18	27011 (12,5V)	128	Intelligent	Quick

PROMs

Bausteinnummer	Baustein	Organisation (in Bit)
1	82 S 123	32*8
2	82 S 129	256*4
3	82 S 135	256*8
4	82 S 131	512*4
5	82 S 141	512*8
6	82 S 137	1024*4
7	82 S 181	1024*8
8	82 S 185	2048*4
9	82 S 191	2048*8
10	82 S 321	4096*8

PALs

Bausteinnummer	Hersteller	Gehäuse (in Pin)
1	MMI	20
2	MMI	24
3	TI	20
4	TI	24
5	NS	20
6	NS	24
7	AMD	20

muß ein Datenblock von AllProg angefordert werden. Der Inhalt dieses 11 Byte langen Blocks hat keine Bedeutung.

Treibersoftware

Bild 23 zeigt die möglichen Parameter-Kombinationen für Befehle. Wer sich nach all diesen Informationen nicht die Mühe machen möchte, den Treiber für mc-AllProg zu schreiben, für den zeigt *Bild 24* eine Sammlung von Pascalroutinen für diese Aufgabe. Sie erfüllt die Anforderungen zum Bedienen von mc-AllProg nach dem oben beschriebenen Protokoll. Es ist im wesentlichen ein Auszug aus der im nächsten Heft beschriebenen kompletten Bediensoftware.

Literatur

[1] *Monolithic Memories*: PAL Handbook (third edition).

[2] *Advanced Micro Devices*: Programmable Array Logic Handbook.

[3] *Texas Instruments*: The TTL Data Book (Volume 3).

[4] *Datenblätter der Firmen Intel, National Semiconductor und NEC.*

[5] VALVO: Datenbuch „Bipolare Speicher, MOS Speicher".

[6] Seng, Peter: mc-EPROM-Brenner, mc 1987, Ausgabe 1, Seiten 46 bis 58.


```
Const (* Protokoll-Bytes *)
 Sync    = $5A; (* Kennzeichnet den Start eines Blocks          *)
 AP_an   = $F0; (* Wird von der Prog.hardware alle 0.5 Sekunden *)
                (* gesendet, nachdem sie eingeschaltet wurde    *)
 OK      = $00; (* Operation OK / Blockanforderung              *)
 Err     = $01; (* Fehler aufgetreten / Fehlerblockanforderung  *)
 TO_aus  = $02; (* Ende der Operation                           *)
Type
 LongInteger= Array[0..2] of Byte;

(* Globale Variablen *)

Var
 Pruefsumme         : Integer;
 Fataler_Fehler     : Boolean;
 BufBlock           : Array[1..127] of Byte;
 Befehl    :
  Record
   BausteinArt : 1..3;        (* 1: PROM, 2: (E)EPROM, 3: PAL *)
   BausteinNr  : Integer;
(* siehe Bild 14 *)
   BefehlsArt  : Byte;
(* Codierung bei PROMs und EPROMs:          *)
(* Bit 7    : 1 => Baustein einlegen        *)
(* Bit 6..0 : 0 => Leertest                 *)
(*            1 => Lesen                    *)
(* nur        2 => Programmieren (I)        *)
(* bei EPROMs:3 => Programmieren (II)       *)
(*                                          *)
(* Codierung bei PALs:                      *)
(* Bit 7    : 1 => Baustein einlegen        *)
(* Bit 6..0 : 0 => Lesen bei 4.5 Volt Vcc   *)
(*            1 => Lesen bei 5.0 Volt Vcc   *)
(*            2 => Lesen bei 5.5 Volt Vcc   *)
(*            3 => Schreiben/Programmieren  *)
   StartAdr    : LongInteger; (* Startadresse *)
  end;

 Abbruch          : Boolean;
 V24_Fehlernummer : Integer;

Procedure Sende_Byte(B : Byte);
begin (* Sende_Byte *)
 BIOS(5, B);  (* Byte B wird zum PUNCHER-Kanal gesendet *)
 Pruefsumme:= (Pruefsumme + B) and $FF;
end;  (* Sende_Byte *)

Procedure Empfange_Byte(Var B : Byte);
begin (* Empfange_Byte *)
 B:= BIOS(6); (* Byte B wird vom READER-Kanal gelesen   *)
 Pruefsumme:= (Pruefsumme + B) and $FF;
end;  (* Empfange_Byte *)

Procedure Fehler_behandeln(FehlerNr: Integer);
begin (* Fehler_behandeln *)
 V24_Fehlernummer:= FehlerNr;
 Fataler_Fehler  := true;
 Sende_Byte(TO_aus);
end;  (* Fehler_behandeln *)

Procedure Empfange_Block(Var Art : Byte; Var Blk);
Var akt_Byte : Integer;
    Block    : Array[1..127] of Byte absolute Blk;
    BuffB    : Byte;
    PS       : Byte;
    Versuche : Byte;
begin (* Empfange_Block *)
(* Synchronisieren *)
 Versuche:= 20;                 (* 20 Versuche, das Sync-Byte zu       *)
 Repeat                         (* empfangen                           *)
  Pruefsumme:= $00;             (* Pruefsumme initialisieren           *)
  Empfange_Byte(BuffB);
  If (BuffB = AP_an) then
   Versuche:= Pred(Versuche);
 until (BuffB = $5A) or (Versuche <= 0); (* bis synchronisiert oder Timeout  *)

 If (Versuche <= 0) then
  Fehler_behandeln(3)           (* ! Timeout-Fehler !                  *)
 else begin
  Empfange_Byte(Art);           (* Blocklaengenbyte empfangen          *)
  For akt_Byte:= 1 to (Art and $7F) do
   Empfange_Byte(Block[akt_Byte]);  (* Datenbytes empfangen             *)
  PS:= Pruefsumme;              (* Pruefsumme fuer Vergleich speichern *)
  Empfange_Byte(BuffB);         (* Pruefsummenbyte empfangen           *)
  If (BuffB <> PS) then
   Fehler_behandeln(1);         (* ! Pruefsummenfehler !               *)
 end;
end;  (* Empfange_Block *)

Procedure Antwort_pruefen;
Var BuffB, BBB, Versuche : Byte;
begin (* Antwort_pruefen *)
 Repeat
  Versuche:= 20;
  Repeat
   Empfange_Byte(BuffB);
   If (BuffB = AP_an) then
    Versuche:= Pred(Versuche);
  until (BuffB <> AP_an) or (Versuche <= 0);
  If (Versuche <= 0) then
   Fehler_behandeln(3)
  else begin
   If (BuffB <> 0) and (BuffB <> AP_an) then begin
    Sende_Byte(Err); (* Fehler-Beschreibung anfordern *)
    Empfange_Block(BBB, BufBlock);
```

Bild 24. Der Kern eines Kommunikations-Programmes nach AllProg-Protokoll für CP/M-80-Rechner

```
    If Not(Fataler_Fehler) then
     Fehler_behandeln(0);
   end;
  end;
 until (BuffB = 0) or Fataler_Fehler;
end;  (* Antwort_pruefen *)

Procedure Sende_Block(Art : Byte; Var Blk);
Var akt_Byte : Integer;
    Block    : Array[1..127] of Byte absolute Blk;
begin (* Sende_Block *)
 Pruefsumme:= $00;
 Sende_Byte(Sync);
 Sende_Byte(Art);
 For akt_Byte:= 1 to (Art and $7F) do
  Sende_Byte(Block[akt_Byte]);
 Sende_Byte(Pruefsumme);
 Antwort_pruefen;
end;  (* Sende_Block *)

Procedure Sende_Befehl(Bausteinwechsel : Boolean);
(* die globale Variable "Befehl" muss entsprechend geladen sein *)
Var Versuche : Integer;
    Ch       : Char;
begin (* Sende_Befehl *)
 Versuche:= 8;
 Repeat
  Fataler_Fehler:= false;
  Sende_Block($80 + 7, Befehl);
  If Fataler_Fehler then begin
   If (BufBlock[1] = 255) then (* Pruefsummenfehler *)
    Versuche:= Pred(Versuche)
   else
    Versuche:= 0;
  end;
 until Not(Fataler_Fehler) or (Versuche <= 0);

 If Not(Fataler_Fehler) and Bausteinwechsel then begin

(*--- Hier muss die Nachricht auf den Bildschirm geschrieben werden, ---*)
(*--- die den Benutzer dazu auffordert, den Baustein in den Program- ---*)
(*--- miersockel zu setzen, und dieses durch das Druecken einer     ---*)
(*--- Taste zu bestaetigen (<ESC> fuehrt zum Abbruch der Funktion).  ---*)

  Read(Kbd, Ch); Abbruch:= (Ch = #27);

(*--- Hier sollte die Nachricht wieder geloescht werden.             ---*)

 end;
end;  (* Sende_Befehl *)

Procedure Empfange_Daten(Anzahl : Integer; Var Zieladresse);
Var BuffB : Byte;
    Anz   : Integer;
begin (* Empfange_Daten *)
 Sende_Byte(OK); (* Datenblock anfordern *)
 Empfange_Block(BuffB, BufBlock);
 If Not(Fataler_Fehler) then begin
  If (BuffB and $80 = 0) then begin
   Anz:= BuffB and $7F;
   If (Anz < Anzahl) or (Anz > 16) then begin
    Fehler_behandeln(2);
   end
  else begin
   Move(BufBlock, Zieladresse, Anzahl);
  end;
 end (* If (BuffB and $80 = 0) *)
 else begin
  Fehler_behandeln(0);

 end; (* If Not(Fataler_Fehler) *)
end;  (* Empfange_Daten *)

Procedure Start_V24;
begin (* Start_V24 *)
 Fataler_Fehler:= false;
 Abbruch:= false;
end;  (* Start_V24 *)

Procedure Stop_V24;

Procedure V24_Error;
begin (* V24_Error *)
 Zeilen_loeschen(4, 6);
 GotoXY(1, 5);
 Write('Fataler Fehler !!!   Befehl wird abgebrochen !  ');
 GotoXY(1, 6);
 Case V24_Fehlernummer of
   0 : begin
        Write(BufBlock[1]:3, BufBlock[2]:4);
        Write(' '); Schreib_Hex(BufBlock[3]);
        Write(' '); Schreib_Hex(BufBlock[4]);
        Write(BufBlock[5]:4);
        Write(' ');
        Schreib_Hex(BufBlock[7]); Schreib_Hex(BufBlock[6]);
        Write(BufBlock[8]:4);
        Write(' ');
        Schreib_Hex(BufBlock[11]); Schreib_Hex(BufBlock[10]);
        Schreib_Hex(BufBlock[9]);
        Write(BufBlock[12]:4);
        Write(BufBlock[13]*50/256:5:1);
        Write(BufBlock[14]*0.1:5:1);
        Write(BufBlock[15]*0.1:5:1);
        Write(BufBlock[16]*0.1:5:1);
       end;
   1 : Write('Pruefsummenfehler !');
   2 : Write('Unzulaessige Datenblocklaenge !');
   3 : Write('Datenblock erwartet !');
  else Write('Unbekannte Ursache !');
 end;
 Write('  ');
 Warte_auf_Taste;
 Zeilen_loeschen(5, 6);
end;  (* V24_Error *)

begin (* Stop_V24 *)
 If Not(Abbruch) then begin
  If Not(Fataler_Fehler) then begin
   Sende_Byte(TO_aus);
  end
  else
   V24_Error;
 end;
end;  (* Stop_V24 *)
```

Josef Ellenrieder

Datenbank-systeme

Teil 3: Wie Sie in die Welt der großen Datenbanken einsteigen können

In den Computer Daten eingeben und wieder abrufen ist eine Sache, auf riesige Datenbestände großer Datenbanken zu greifen eine andere – aber eine sehr interessante.

Ein Personal Computer kann den Anwender komfortabel mit Frage-Antwort-Abläufen (Menütechnik) durch ein Programm führen. Er kann in umfangreichen Auswertungen oder Statistiken gezielt die falschen Ziffern herauspicken und berichtigen. Er kann aber auch mit erstaunlich anspruchsvollen Programmen arbeiten: über schlichte Telefonleitungen in Verbindung mit Großrechnern in anderen Städten und anderen Kontinenten. Es ist erstaunlich, wie menschliche, ungebundene Kreativität in verschiedenen Unternehmen unterschiedliche Verfahrensweisen, Formulare oder Abläufe zum gleichen realen Geschäftsvorfall entwickelt. Datenbanken und Programmsysteme auf unterschiedlichen Rechnern entwickelten sich in einer ihnen jeweils eigenen und einzigartigen Dynamik. Überspitzt ausgedrückt, die einzige Gemeinsamkeit zwischen ihnen ist der Tatbestand, daß Daten gespeichert sind und daß Programme existieren. Daten aus unterschiedlichen Quellen in einer einheitlichen, für die eigenen Zwecke verständlichen Form zusammenzustellen ist häufig mit erheblichem Aufwand verbunden. Hier begannen die Probleme des Organisators und des Anwenders.

Kommunikation nach internationaler Norm

Inzwischen hat sich die Fachwelt auf international genormte Kommunikationsprotokolle geeinigt. Ähnlich dem Code des Morsealphabets mit langen und kurzen Signalen unterhalten sich Großrechner untereinander, aber auch Großrechner und Personal Computer. Die Kommunikation zwischen zwei Partnern besteht nicht allein im Artikulieren von Lauten, sondern man muß sich eines beiden Seiten bekannten Wortschatzes bedienen.

Wenn der Benutzer einem Personal Computer die Kommunikationsgrammatik, also die erlaubten Worte bei der Rechnerunterhaltung (Beispiele: Akzeptiert, Warten, Fehler, Neuer Block, Senden) einmal vermittelt hat, kann er mit dem Großrechner in Verbindung treten. Was man benötigt, um beide zusammenkommen zu lassen, sind im allgemeinen zwei Nummern. Die erste Zahl ist die lokale Telefonnummer für einen Datenkonzentrator. In der BRD ruft der Kommunikationssuchende beispielsweise eine Nummer des Datex-Dienstes der Bundespost an. Dem Datex-System teilt er dann mit, wen er am anderen Ende der Leitung sprechen möchte. Ist die Verbindung aufgebaut, so erscheint auf dem Bildschirm im allgemeinen die Aufforderung, sich zu identifizieren.

Wenn dem Personal Computer ferner alle Antworten auf Fragen des fremden Computerprogramms vorgegeben sind, kann er sich automatisch mit dem fremden Programm „unterhalten". Er kann so zum Beispiel den Rechner veranlassen, daß die Verkaufsdaten des Modells „ABC" über die Leitung übertragen und auf dem Personal Computer gespeichert werden.

Jeder PC sollte eine „eingebaute" Uhr und einen Kalender besitzen. Damit stehen faszinierende Möglichkeiten der Bürokommunikation und lokalen Datenverarbeitung offen. Was vielleicht noch beeindruckender ist: Das alles ist keine graue Theorie.

Der PC wählt sich z. B. zu festen Zeitpunkten an fremde Rechnersysteme an. Wenn der Anruf von den Prüfungsmechanismen akzeptiert wird, veranlaßt das PC-Programm als nächsten Schritt den Großrechner zum Übertragen dort vorhandener Daten. Danach bricht der PC die Verbindung mit dem fremden Rechner ab und wählt einen anderen Rechner an. Von dort werden dann weitere Daten „eingesammelt", andere Programme genutzt. Danach beginnt automatisch die Auswertung der gesammelten Daten. Das alles kann geschehen, ohne daß die Anwesenheit des Benutzers notwendig ist. Er hat zuvor ja alle notwendigen „Nummern" auf der Diskette des PC gespeichert.

Alle Arbeitsabläufe sind programmgesteuert und wurden bei einer maßge-

DATENB.-NAME	HERSTELLER	SACHGEBIETE	ONLINE-ANBIE
ABDA-PHARMA	Dt.Apothekerverband, Frankfurt	Arzneistoffe, Arzneihersteller	DIMDI, Köln, Tel. 0221-47241
BIBLIO-DATA	Dt. Bibliothek, Frankfurt	Fachbücher, Zeitschriften	INKA, 7514 Eggenstein/Leopold.
BOOK	Börsenverein d.Dt.Buchhandels	Bücher, Verlagsadressen	RZ Heinze, Tel. 05141/500
BOOK	Grossohaus Wegener	Bücher, Verlagsadressen	RZ Heinze, dto.
BOOK	Novotron, 4600 Dortmund	Videos, Bezugsquellen	RZ Heinze, dto.
DEQUIP	DECHEMA, 6 Frankfurt	Anlagen, Apparate, Maschinen	STN Intern. 75 Karlsruhe
DEQUIP	FIZ Chemie, 1 Berlin	Biotechnologie	STN Intern., 75 Karlsruhe
DETEQ	DECHEMA, 6 Frankfurt	Hilfsmittel f. chem. Technik	STN Intern., 75 Karlsruhe
BDI	Sachon-Verlag, Mindelheim	Produktions- u. Konsumgüter	FIZ Technik, 6 Frankfurt
BDI	Sachon-Verlag, Mindelheim	Dt. Hersteller f. dt. Produkte	GENIOS-Wirtschaftsd. Düsseld.
AMPRO	Hoppenstedt-Datenbank, Darmst.	Produkte Maschinenbau, Elektro	FIZ Technik, Frankfurt
MRA	Hoppenstedt-Datenbank	Produkte Meß- u. Regeltechnik	FIZ Technik, Frankfurt
VDMA	VDMA, Frankfurt	Produkte Maschinenbau u. Herst	FIZ Technik, Frankfurt
ISIS	Nomina, München	Software-Produkte	FIZ Technik Frankfurt
SENSOR	FIZ-TECHNIK, Frankfurt	techn.Sensoren u. Hersteller	FIZ Technik, Frankfurt
DAFID	DAFID, 2808 Syke	Meßeinrichtungen Umweltschutz	unbekannt
BRS Info	Bertelsmann Informationsservic	Wirtschaftsinformationen	Bertelsmann, München
Juris	Juris GmbH, Bonn	Rechtsinformationen	DATEV, Nürnberg

Bild 14. Datenbanken, die in der Bundesrepublik Deutschland erreichbar sind

Infovermittl	Anschrift	Tel.
Proconsult Beratung	8900 Augsburg	0821-404433
TBR der IHK Bochum	463 Bochum	0234-60396
FIZ Technik	6 Frankfurt	0611-43081
TT-Technologietrans.	69 Heidelberg	05221-28212
Online GmbH	69 Heidelberg	06221-21536
GID Informationszen.	6 Frankfurt	0611-6687-36
INKA	7514 Eggenstein	07247-824600
IHK Koblenz	5400 Koblenz	
Dr.Schulte-Hillen	5 Köln 71	0221-795091
IRS Info-Institut	8 München	089/263060

Bild 15. Informationsvermittlungsstellen, die beim „Online-Suchen" behilflich sind

schneiderten Anwendung im Zusammenspiel zwischen Analytiker, Programmierer und Anwender festgelegt. Bei Programmpaketen „von der Stange" wird der Arbeitsablauf von den Entwicklern des Pakets nach bestem Wissen vorgedacht. Wenn der Anwender morgens das Büro betritt, hat sein Personal Computer diese gesamte Arbeit schon getan. Er läßt sich dann nur noch die Ergebnisse ausdrucken.

Suche in großen Dateien dauert lange

Die Vorteile der Kooperation von PC und Großrechner: Verwaltung, Pflege, Speicherung und Abfrage großer Datenbestände sind immer noch Domäne der Großrechner. Besonders bei umfangreichen Suchvorgängen in großen Datenbeständen tut sich der Personal Computer noch schwer und braucht viel Zeit. Der Verbund zentraler Datenspeicherung und dezentraler Verarbeitung erhält mit der „Intelligenz" des Personal Computers ein großes Gewicht.

Hingewiesen sei nur auf die Auskunftssysteme in Reisebüros, Versicherungen und Banken. Auch „Electronic Mail"-Systeme gehören zu dieser Anwendungskategorie. Diese Systeme wurden erst mit der Intelligenz vor Ort möglich. Umfangreiche ablauforganisatorische Probleme in den EDV-Bereichen entfallen fast ersatzlos, da jeder berechtigte Anwender direkten Zugang zu den Daten erhalten kann.

Großrechner und PC ergänzen sich

So ergänzen sich Großrechner und PC in hervorragender Weise. Der Großcomputer übernimmt alle umfangreichen, zeitaufwendigen, an Spezialeinrichtungen gebundenen „zentralen" Aktivitäten. Damit entlastet er massiv die lokale Datenverarbeitung, die sich auf die spezifisch lokalen Bedürfnisse konzentrieren kann. Lokal arbeitet es sich somit deutlich effektiver.

Ähnliches gilt für den Großrechner. Er wird von allen „den Dienstbetrieb" störenden Kleinarbeiten entlastet. Jeder Anwender sammelt über den PC die für die weitere lokale Verarbeitung notwendigen Daten mit seinen eigenen spezifischen Programmen unter seiner Kontrolle und Verantwortung.

Man kann daher ohne Einschränkung die Feststellung treffen, daß die Einführung des Personal Computers das Tagesgeschäft deutlich vereinfachen und erleichtern kann.

Wo bereits kleine und mittlere Unternehmen in der Bundesrepublik eine Online-Recherche sinnvoll durchführen können, sagt Ihnen *Bild 14*. Welche Informationsvermittlungsstellen Ihnen beim „Online-Suchen" behilflich sind, zeigt *Bild 15*.

Cursor-Modus beim mc-68000-Computer

Das folgende C-Programm steuert den Cursor-Anzeigemodus des mc-68000-Computers. Übergabeparameter beim Aufruf mit cursor(string) ist ein String, der in Anführungszeichen eingeschlossen sein muß. Er bezeichnet den gewünschten Cursor-Modus, man braucht sich also keine Zahlen oder Codes zu merken. Taucht hier ein nicht zugelassener Modus auf (siehe Bild), so wird er ignoriert und der Cursor in Normalform gebracht.
Verwendet wurde die CLIB-Bibliotheksroutine _strcmp. Die Zeiger (auf Bytes) pm, pn und pe dienen folgenden Zwecken:

pm spricht das Cursor-Modus-Register des CRTC an,
pn lädt das so selektierte Register mit dem geforderten Wert,
pe lädt das Monitor-Cursor-Register mit dem gleichen Wert – wenn das nicht geschieht, läuft nichts!

Im Cursor-Modus-Register dürfen die Bits 5 und 6 nur durch ein logisches Oder gesetzt werden, da die anderen Bits dieses Registers (Nr. 10) die Cursor-Höhe bestimmen. Dazu sei auf das im Handbuch des mc-68000-Computers mitgelieferte Datenblatt hingewiesen. Es ist sinnvoll, diese Routine nach dem Compilieren mit dem Archivprogramm-Aufruf ar68 rav clib cursor.o in die CLIB-Bibliothek einzubinden, damit sie anderen Programmen jederzeit zur Verfügung steht. *Reinhard Boenigk*

```
int cursor(string)
char *string;
{ char *string1,*string2,*string3,*string4,*pm,*pn,*pe;
  int _strcmp();
  string1 = "aus";                  /*Dies sind die moeglichen Uebergabeparameter.*/
  string2 = "normal";
  string3 = "schnell blinken";
  string4 = "langsam blinken";
  pm = 0xFE4000;       /*CRTC ansprechen: bei Busadresse BA1 = 0 wird das ent-*/
  pn = 0xFE4002;  /*sprechende Register selektiert; bei BA1 = 1 wird es ueber*/
  pe = 0xFFE7EF;                                  /*den oberen Datenbus geladen.*/
  if (!_strcmp(string,string1))                         /*Cursor-Modi: hier: aus*/
     { *pe = 0x20; *pm = 0x0A; *pn != 0x20; }
  else if (!_strcmp(string,string2))                                /*hier: normal*/
        { *pe = 0x00; *pm = 0x0A; *pn != 0x00; }
  else if (!_strcmp(string,string3))                      /*hier: schnelles Blinken*/
        { *pe =  0x40; *pm = 0x0A; *pn != 0x40; }
  else if (!_strcmp(string,string4))                      /*hier: langsames Blinken*/
        { *pe = 0x60; *pm = 0x0A; *pn != 0x60; }
  else { *pe = 0x00; *pm = 0x0A; *pn != 0x00; }     /*hier: default bei unzu-*/
}                                                  /*laessigen Eingaben: normal*/
```

Änderbare Cursor-Darstellung in C: Da der Baustein 6845 auch in anderen Rechnern zum Einsatz kommt, läßt sich diese Routine in abgewandelter Form auch dort verwenden

men und in Basic kein wesentlicher Geschwindigkeits-Unterschied zwischen den drei getesteten ATs.

Dokumentation: Große Unterschiede

Die mit Abstand beste Dokumentation kann Osborne vorweisen. Sie ist deutschsprachig, rund 200 Seiten stark und beschreibt im Gegensatz zum (englischen) Tulip-Handbuch nicht nur elementare DOS-Befehle, sondern auch die Funktionsweise des AT, seine Adressenbelegung und die Bedeutung der einzelnen DIP-Schalter und Brücken auf der Platine.
Demgegenüber ist die MCI-Dokumentation zwar auch deutschsprachig, umfaßt aber nur zehn Seiten und enthält lediglich eine Liste der von MCI verfügbaren Ausbaumöglichkeiten. Wem das nicht genügt, muß sich zudem damit abfinden, daß auch eine telefonische Hilfestellung nur begrenzt von MCI erwartet werden darf – das wissen wir nicht nur von Lesern, sondern haben wir auch selbst erfahren müssen. Die einzige mitgelieferte Diskette, die lediglich von Hand mit „S" beschriftet war, stellte sich als unlesbar heraus und sollte wohl die Systemkonfiguration durch Schreiben in das batteriegepufferte CMOS-RAM erlauben; das auf der Osborne-Utility-Disk vorhandene Programm SETUP funktionierte dann aber auch auf dem MCI-AT. (Leider dürfte kaum ein MCI- oder Tulip-Kunde das Osborne-Programm zur Verfügung haben ...)
Und die Preise? Am tiefsten muß man für den kleinsten AT in die Tasche greifen, für den Tulip also. Er kostet in einer typischen Konfiguration rund 50 % mehr als der MCI-AT. Mit dem Osborne-AT II als Bausatz kommt man verständlicherweise dabei noch ein wenig günstigster weg. Zum Glück muß man dabei nicht erst Platinen bestücken – diese werden fertig und getestet geliefert, die nötigen Arbeiten beschränken sich faktisch auf den Einbau ins Gehäuse, was vergleichsweise problemlos ist.
Abschließend sei bemerkt, daß alle Computer, gleich welchen Herstellers, laufend technischen und preislichen Änderungen unterworfen sind. Ein Vergleich wie der hier gemachte läßt also lediglich Rückschlüsse auf den Zustand an einem bestimmten Stichtag zu – vom Preis des Tulip-AT bis hin zur versprochenen, aber fehlenden Taktfrequenz-Umschaltbarkeit des MCI-AT, die in den zwei Monaten zwischen der Druchführung des Tests und seinem Abdruck vermutlich schon eingebaut wurde.

Siemens PC-D mit NDR-Klein-Bus

Ohne Veränderungen an der Hardware kann der Siemens PC-D zur Ansteuerung des NDR-Klein-Busses verwendet werden. Dazu wird ein meist nicht verwendeter Platinensockel auf der Hauptplatine des Siemens PC-D benutzt. Durch die hier beschriebene Beschaltung des Sokkels D 143 kann man so die zahlreichen Karten, die es für den NDR-Klein-Computer gibt, am Siemens PC-D einsetzen. Auf der Hauptplatine (S26361-D270) kann der Sockel D 143 als Anschluß an den Systembus verwendet werden. Die Pinbelegung ist in *Bild 1* aufgeführt. Der Sockel ist im Adreßbereich von F000H...F7FFH anzusteuern. Mit einem 24poligen DIL-Stecker mit Flachbandkabel werden die Signale aus dem Siemens PC-D wie in *Bild 2* dargestellt auf den NDR-Klein-Bus geleitet. In *Bild 3* schließlich haben wir ein kleines Beispielprogramm in Turbo-Pascal zur Ansteuerung einer I/O-Karte wiedergegeben. Neben verschiedenen I/O-Karten für Steuerungen wurden mit dem EPROM-Programmiergerät PROMER bereits EPROMs programmiert und mit der Grafikkarte GDP64 externe Monitore angesteuert.

Dipl.-Ing- (FH) Manfred Weis

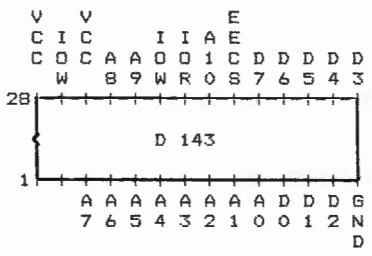


Bild 1. Die Pinbelegung des 28poligen Sockels D 143 auf der Hauptplatine des Siemens PC-D

Pin D143	Signal	NDR-KLEIN-BUS	Pin D143	Signal	NDR-KLEIN-BUS	
1		--	15	D3	11	
2		--	16	D4	12	
3	A7	27	17	D5	13	
4	A6	26	18	D6	14	
5	A5	25	19	D7	15	
6	A4	24	20	EECS	18	(IORQ)
7	A3	23	21	A10	36	
8	A2	22	22	IOR	16	(RD)
9	A1	21	23	IOW	17	(WD)
10	A0	20	24	A9	35	
11	D0	8	25	A8	34	
12	D1	9	26	VCC	4,5	
13	D2	10	27		--	
14	GND	6,7	28		--	

Bild 2. Nach diesem Schema werden die Signale des Siemens PC-D an den NDR-Klein-Rechner übergeben

```
program bus_test;

const in0,out0 = -4096; { F000h }
      in1,out1 = -4095; { F001h }
begin
  repeat
    portw[out0]:=170;        {     1  0  1  0  1  0  1  0      }
    delay(5000);
    portw[out0]:= 85;        {     0  1  0  1  0  1  0  1      }
  until keypressed = true;
end.
```

Bild 3. Beispiel zur Ansteuerung einer I/O-Karte

Dieter Strauß

CAD für den PC

Schaltpläne zeichnen mit Shamrock-CAD

Vor nicht allzu langer Zeit liefen CAD-Programme nur auf großen Computern. Mittlerweise gibt es zahlreiche CAD-Programme für den PC. Ein besonders preiswertes Programm, das den Vergleich mit teueren Produkten nicht scheut, ist Shamrock-CAD. mc hat dieses Produkt auf einem PC und einem AT getestet. Shamrock-CAD wird mit umfangreichen Schaltzeichen-Bibliotheken geliefert und ist besonders gut zum Zeichnen von Schaltplänen geeignet.

Shamrock-CAD stellt zum Zeichnen die Grundelemente Linie, Kreis, Kreissegment und Text zur Verfügung. Das Programm wird zusammen mit einem Bediener-Handbuch, einem Bibliotheks-Handbuch und fünf Disketten geliefert. Allein die Bauteile-Bibliothek mit über 400 der gängigsten TTL-Symbolen ist auf drei Disketten untergebracht. Das eigentliche CAD-Programm und die Utilities zum Ausdrucken und Plotten der Zeichnung sind auf je einer Diskette. Das Bediener-Handbuch ist in zwei Teile gegliedert: Tutorial und Nachschlagewerk.
Im Tutorial wird am Beispiel eines einfachen Schaltplans der Anwender schrittweise in die Bedienung des Programms eingeführt. Alle im Tutorial gezeigten Beispiel können vom Anwender sofort am System nachvollzogen werden. Der zweite Teil des Handbuchs ist als Nachschlagewerk gedacht. Im Bibliotheks-Handbuch sind sämtliche Schaltzeichen der mitgelieferten Bibliotheken abgedruckt.
Wir haben das Programm auf dem Commodore PC 10 und dem MCI AT4SLC getestet.
Der Commodore PC 10 wurde dazu in folgender Konfiguration betrieben:
– 640 KByte RAM
– Hercules-kompatible Grafik-Karte

Es hat sich gezeigt, daß bei geschickter Installierung auch ohne Festplatte sehr schnell mit Shamrock-CAD gearbeitet werden kann. Eine RAM-Floppy mit 256 KByte macht's möglich! Die in *Bild 1* gezeigte Installation durch eine Stapeldatei hat sich beim PC 10 während des Tests bewährt. Mit der RAM-Floppy bleibt dem Anwender der nervenraubende langsame Floppy-Zugriff erspart.

Beim AT-kompatiblen Rechner MCI AT4SLC kann auf die RAM-Floppy verzichtet werden, da die Festplatte schnell genug ist.
Obwohl das Fadenkreuz während des Zeichnens mit den vier Pfeiltasten der Tastatur auf jede Stelle der Zeichenfläche positioniert werden kann, sollte sich jeder ernsthafte Anwender eine Maus zulegen. Bewährt hat sich in unserem Test die Lingo-Maus. Diese Maus wird einfach an der V.24 – Schnittstelle des PCs angeschlossen und mit einem Zwischenstecker in die Tastaturzuleitung eingeschleift. Da der Commodore PC 10 den Tastaturanschluß an der Vorderseite des Systemgehäuses hat, muß die Leitung der Lingo-Maus, die zum V.24-Anschluß des PC 10 hinführt, etwas verlängert werden.
Nach der Installierung eines Maus-Treibers, der kompatibel zur Microsoft-Maus sein muß, lassen sich nach dem Start des CAD-Programms fast alle Eingaben, außer der Benennung von Symbolen, Dateien usw., mit der Maus durchführen. Der Bildschirm ist in eine Zeichenfläche und eine Menüleiste (am rechten Bildschirmrand) aufgeteilt. In der Menüleiste werden der Maßstab, die Position des derzeitigen Bildausschnitts in der gesamten Zeichenfläche und die Koordinaten des Fadenkreuzes angezeigt.

Zwischen den beiden Hauptmenüs kann der Anwender mit der Esc-Taste hin- und herschalten. Alternativ dazu kann er mit der rechten Maustaste das mit „Esc" bezeichnete Feld der Menüleiste anklicken. Von Untermenüs aus werden die verschiedenen Funktionen aufgerufen.
Hilfstexte können jederzeit durch Eingabe eines Fragezeichens oder durch Anklicken des Help-Feldes angefordert werden.
Nach dem Programmstart (Aufruf mit CAD) gelangt der Anwender sofort in das Workfile-Menü. Mit der Funktion „Make" wird der Name und der Zugriffspfad der Arbeitsdatei spezifiziert. Als nächstes sollte das Untermenü Parameter aufgerufen werden. Die Funktion „Install" dient zur Eingabe der Abmessungen des aktiven Zeichenfeldes. Dies ist sehr wichtig, damit die zukünftige Zeichnung in definierten Maßstäben auf dem Bildschirm verzerrungsfrei wiedergegeben werden kann.

Während unseres Test hat es sich als vorteilhaft erwiesen, der Zeichnung immer ein Hilfsraster als Orientierungshilfe zu unterlegen. Mit der Funktion F2 (R-Dista) wird der Rasterabstand des Hilfsrasters definiert. Beim Zeichnen von einfachen Schaltplänen reicht 1 mm meistens völlig aus. Nur bei einer guten Grafikkarte, wie z. B. der Hercules-Karte,

```
ECHO OFF
keybgr
msmouse
verify on
ECHO RAM-DISK MIT 256 KBYTE AUF C: INSTALLIERT
copy cad.com c:
copy cad.000 c:
copy cad.hlp c:
copy cad_text.msg c:
copy cad_4X6.FON c:
copy cad_8X8.FON c:
copy cad_14X9.fon c:
c:
cad
echo Bitte Diskette für Sicherungskopie in B: einlegen!
Pause
copy c:work.pix b:
Echo on
```

Bild 1. Eine Stapeldatei und eine RAM-Floppy mit 256 KByte erhöhen den Bedienungskomfort

erscheint bei einem Rasterabstand von 1 mm und dem Maßstab 4:1 ein Raster auf dem Bildschirm. Sobald die Entfernung zweier Rasterpunkte den Mindestabstand unterschreitet, der durch die maximal mögliche Auflösung der verwendeten Grafik-Karte festgelegt ist, erfolgt keine Ausgabe des Rasters. Wird das Hilfsraster zum Zeichnen gebraucht, so muß man den Maßstab vergrößern. Dies geht natürlich nur auf Kosten des Bildauschnittes, d.h., mit zunehmenden Maßstab wird der Bildauschnitt kleiner. Mit der Funktion F4 (Raster) kann der Anwender das Raster manuell ein- oder ausschalten.

Stücklisten einfach erstellt

Mit Shamrock-CAD lassen sich Schaltpläne mit Stück- und Verbindungslisten erstellen. Die Verbindungsliste kann vom weitverbreiteten Autorouter 3 zum Zeichnen eines Platinen-Layouts verwendet werden.

Damit ein Computer eine Stück- und eine Verbindungsliste aus einem Schaltplan erstellen kann, muß der Anwender bereits beim Eingeben eines Schaltplans markieren, welche Linie eine Zeichenlinie und welche eine Verbindungslinie ist. Eine Verbindungslinie – auch Signalleitung genannt – verbindet die einzelnen Anschlüsse der verschiedenen Bauteile. Sie besteht aus einem Linienzug mit einem definierten Anfangs- und Endpunkt. Shamrock-CAD kennzeichnet die Anfangs- und Endpunkte einer Signalleitung mit einem Flag. Ein Flag besteht aus einer Anschlußbezeichnung mit maximal 3 Zeichen, einer Elementbezeichnung mit bis zu fünf Zeichen und aus einem Kommentar mit maximal 36 Zeichen. So könnte ein Flag in einer Zeichnung lauten:

Anschlußbezeichnung: 12
Elementbezeichnung: IC4
Kommentar: 74LS138

Die Funktion F1 (Display) im Untermenü „Flags" schaltet je nach bisherigen Anzeigezustand die Darstellung der Flags ein- oder aus. Damit kann man sich schnell einen Überblick über bereits verwendete Bezeichnungen verschaffen.

Franzis-CAD auf einen Blick:

Hardware-Voraussetzungen:

- IBM-PC/XT/AT oder kompatible Computer
- 256 KByte RAM (512 KByte empfehlenswert)
- Grafik-Karte

Verwendbar sind:

CGA-Karte (640 × 200 Punkte)
Hercules-Karte (720 × 340 Punkte)
Olivetti-Grafik (640 × 400 Punkte)

Software-Voraussetzungen:

- MS-DOS ab 2.0
- Maustreiber (kompatibel zur Microsoft-Maus)

Merkmale:

Auflösung: 0,01...0,2 mm (wählbar)
Zeichenfläche: 32767 × 32767 Punkte
bei 0,01 mm Auflösung: 0,32 m × 0,32 m
bei 0,2 mm Auflösung: 6,4 m × 6,4 m
Maßstab: 1:3...16:1

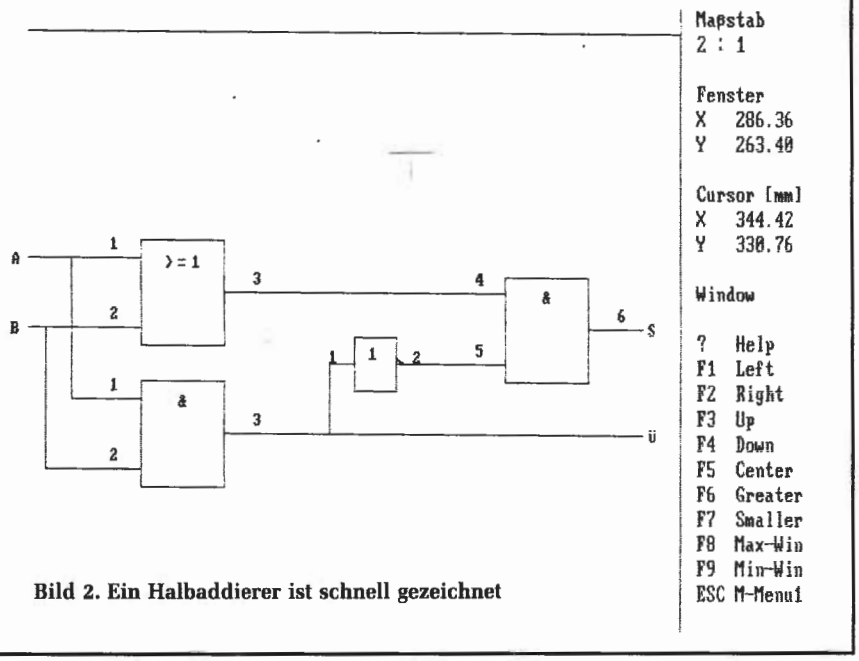


Bild 2. Ein Halbaddierer ist schnell gezeichnet

Wie wird eine Linie gezeichnet?

Im Untermenü „Line" stehen mehrere Funktionen zum Zeichnen von Linien zur Verfügung. Der Anfangspunkt einer Linie wird durch Anklicken der Funktion F1 (From) auf der Menüleiste oder durch Betätigen der Funktionstaste F1 markiert. Mit der Funktion F3 (To) wird das Ende eine Linie markiert. Waagerecht/senkrechte Linienkombinationen können schnell mit der Funktion F2 (Over) gezeichnet werden. Um das Zeichnen von Linien zu erleichtern, sollten Sie immer an einem der Hilfsrasterpunkte beginnen bzw. aufhören. Denn wer schafft es schon bei einer waagrechten Linie, „freihändig" den Anfangs- und Endpunkt in genau der gleichen Y-Koordinate mit der Maus zu markieren? Um schnell das Fadenkreuz auf einem Rasterpunkt zu positionieren, bewegt man das Fadenkreuz mit der Maus nur bis in die unmittelbare Nähe des Rasterpunktes und drückt die linke Maustaste oder die Enter-Taste. Darauf wird das Fadenkreuz exakt auf den Rasterpunkt positioniert.
Die beschriebenen Funktionen zum Zeichnen einer Zeichenlinie werden auch zum Zeichnen von Signalleitungen verwendet. Die Umschaltung zwischen Zeichenlinien und Signalleitungen geschieht mit der Funktion F9 im Untermenü „Line". In der Kopfzeile erscheint die Meldung „Signal", wenn der Modus zum Zeichnen von Signalleitungen aktiv ist.
Eine Signalleitung kann, insbesondere wenn es sich um eine Bus-Leitung handelt, mit sehr vielen Schaltungselementen verbunden sein. Eine Signalleitung muß daher nach der Verlegung ihres letzten Teilstücks dem System als abge-

```
ANALYSE.COM:    Analysiert den Schaltplan.
CAD_INST.MES:   Hilfsdatei für CAD_INST.COM
CAD_INST.COM:   Installationsprogramm für den Plotter-Treiber
COMPRESS.COM:   Restauriert beschädigte Bibliotheks-Dateien
CONVERT.COM:    Konvertiert die Arbeitsdatei in eine Textdatei
DRUCKER.COM:    Gibt die Zeichnung über einen Drucker aus
HP-GL.DAT:      Plotter-Treiber für PP/GL-kompatible Plotter
MP1003.DAT:     Plotter-Treiber für Graphtec-Plotter MP1003
MP2000.DAT:     Plotter-Treiber für Graphtec-Plotter MP1003
ONE_PEN.COM:    Vereinheitlicht die gespeicherten Strickstärken
PLOTTER.COM:    Ausgabe der Zeichnung über einen Plotter
SETCOM1.COM:    Installiert die serielle Schnittstelle
SYMBOLS.COM:    Erstellt die Liste aller Symbole einer Bibliothek
```

Bild 3. Utilities erleichtern die Anwendung von Shamrock-CAD

schlossen mitgeteilt werden. Dazu muß die Funktion F3 (To) zweimal aufgerufen werden. Nach dem Abschluß der Signalleitung verschwindet der Hinweis „Signal active" in der Kopfzeile.
Sowohl im Signalleitungs-Modus als auch im Zeichenlinien-Modus stellt das Programm eine Funktion zum punktgenauen Ansetzen an bereits vorhandene Linien bereit.
Sehr gut gefällt uns, daß es mit Shamrock-CAD auch möglich ist, Linien zu zeichnen, ohne das Untermenü „Line" aufzurufen. Jederzeit kann der Anwender mit der „."-Taste den Beginn einer Linie und mit der „,"-Taste das Ende einer Linie markieren. Die Arbeitsgeschwindigkeit läßt sich erheblich steigern, wenn die linke Hand zur Bedienung der Maus und die rechte Hand für Tastatureingaben benutzt wird. Shamrock-CAD stellt übrigens noch einige andere Funktionen zur Verfügung, um das lästige „Blättern" in Menüs zu vermeiden.
In *Bild 2* wird die Schaltung eines Halbaddieres dargestellt. Dieses Bild wurde nach dem Betätigen der PrtSc-Taste auf einem Matrix-Drucker ausgegeben.

Selbstverständlich enthält die Utility-Diskette einen Plotter-Treiber, der an praktisch jeden Plotter anpaßbar ist. *Bild 3* zeigt die Liste der mitgelieferten Utilities.

Shamrock-CAD ist in zwei Versionen lieferbar:

– für PCs mit CGA- oder Hercules-Karte
– für Olivetti M24/M21/M28L

Nach unserem Test sind wir der Meinung, daß Shamrock-CAD für viele PC-Benutzer, die Schaltpläne zeichnen und Stücklisten erstellen wollen, eine preiswerte Alternative zu wesentlich teueren Programmen ist.

Taiwan schlägt zu

Nach der Vorstellung von IBMs neuer PC-Familie PS/2 glaubten viele, daß damit das Ende der Clone-Hersteller gekommen sei. Doch Taiwan reagiert schneller als es IBM lieb sein kann. Voraussichtlich wird Mitac der erste Hersteller sein, der das Modell 30 nachbaut. Experten glauben, daß Mitac schon vor der offiziellen Ankündigung der PS/2-Familie detaillierte Unterlagen über die Mikrokanal-Architektur hatte. Auch die Software des BIOS scheint keine allzu großen Probleme zu bereiten. Staatlich unterstützte Forschungsorganisationen entwickeln bereits kompatible BIOS-Versionen. ERSO, die bekannteste Forschungsorganisation, ist übrigens in unmittelbarer Nähe von Mitec ansässig. Mitac fertigt die gesamte PC-Linie vom XT bis zum 386er-System. Dazu gehört auch der Paragon 88, ein bereits mit Bausteinen in Oberflächenmontage-Technik (SMT) bestückter PC mit der 8088-CPU. Insider des PC-Marktes haben festgestellt, daß viele Hersteller die OEM-Version des Paragon 88 kaufen und ihn unter einem anderen Namen auf den Markt bringen. St

Altos mit 80386

Einen neuen Unix-Rechner mit Intels 80386 und bis zu 16 MByte RAM, der bis zu 64 Arbeitsplätze bedient, stellte die Firma Altos vor. Eindeutig schneller als das bisherige Altos-Flagschiff mit einem 68020 ist dieser Computer aber nicht unbedingt wegen des Intel-Prozessors, sondern wegen einer raffinierten Cache-Technik, durch die der Prozessor von allen Nebenaufgaben der virtuellen Speicherverwaltung entbunden wird. Auch Mehrprozessor-Betrieb wird von der Speicherverwaltungs-Hardware unterstützt – doch will man bei Altos noch auf die gemeinsame Unix-Version von Microsoft und AT&T warten, bevor man einen entsprechenden Rechner vorstellt. Dieses Betriebssystem wird dann nämlich auch die Software-Seite des Mehrprozessor-Betriebes sichern. Der nachträgliche Einbau einer zweiten CPU-Karte dürfte aber kein Problem sein.
Bis dahin arbeitet das System unter Unix/Xenix V, das nun auch für die älteren Altos-Rechner mit 68020 und 80286 angeboten wird. In einem „Emulation Mode" wird jedoch auch Unix-III-Software problemlos verarbeitet.
Das Prozessor-Rennen jedenfalls geht weiter: Wie man hört, soll ein 68030-Rechner bei Altos schon in der Entwicklung sein ...

Nach Unterlagen der Firma Altos

„Die Angst

vor einem Mißbrauch der Volkszählungsdaten sei jedoch unbegründet. Anders, als mancherorts vermutet, werden die gewonnenen Daten nämlich nicht in Computern, sondern auf Magnetbändern und Platten gespeichert. "

Oberregierungsrat Lorenz, neutraler Referent auf einem Informationsabend in Markt Wolnzach;
zitiert nach einem Bericht im Pfaffenhofener Kurier/Wolnzacher Anzeiger vom 28. April 1987.

Rüdiger Bäcker, Willi Sicking

Uhrenbaustein RTC-58321

Eine Uhrenkarte für den NDR-Klein-Computer

Mit dem RTC-58321 steht ein preiswerter und flexibler Uhrenbaustein zur Verfügung. Der Anschluß an einen Rechner über eine Ein-/Ausgabe-Schnittstelle soll am Beispiel des NDR-Klein-Computers demonstriert werden.

Nachdem der ursprünglich auf der Uhrenkarte des NDR-Klein-Computers eingesetzte Baustein E05-16 nicht mehr hergestellt wird, stellt sich die Frage nach einer Alternative. Mit dem RTC-58321 ist eine Software-Emulation der Orginalkarte möglich; die E05-Karte kann daher problemlos ersetzt werden. Die Übertragung auf andere Rechner ist ebenfalls leicht möglich.
Die Schaltung beinhaltet eine Überraschung im positiven Sinn: Man braucht keine zusätzliche Karte! Der RTC-58321 wird einfach mit auf die IOE-Karte gesetzt, auf der sich schon die Centronics-Schnittstelle befindet. Dazu braucht man nur zwei vierfach Tristate-Treiber, zwei Widerstände, zwei Dioden, einen Taster und eine Lithium-Batterie. Die in der Schaltung verwendeten Inverter sind auf der IOE-Karte noch frei.

Alles an Bord

Aus dieser Aufzählung ist ersichtlich, daß in dem verwendeten Uhrenbaustein schon alles mögliche „drin" sein muß. Daher zunächst eine kurze Beschreibung des Bausteins:
Der RTC ist ein Echtzeit-Modul mit integriertem Quarzoszillator, Backup-Schaltung für Akku-Betrieb, Zeit- und Datumsfunktion sowie einem Taktausgang für Interrupt-Betrieb. Verpackt ist das Ganze in in einem 16poligen DIP-Gehäuse. Durch den eingebauten Quarz ist kein Abgleich erforderlich, was einfachen Nachbau der Schaltung sichert. Der Baustein besitzt einen eingebauten Zeit- (Stunden, Minuten, Sekunden) und Datumszähler (Jahr, Monat, Tag und Wochentag). Man kann zwischen 12- und 24-Stunden-Anzeige umschalten, auch Schaltjahre werden automatisch berücksichtigt. Auf den Datenleitungen können Impulse mit einer Periodendauer von 1/1024 Sekunde, 1 Sekunde, 1 Minute oder 1 Stunde ausgegeben werden. Des weiteren ist eine Start-Stop-Reset-Funktion vorhanden.
Zum Anschluß des RTC wird nur ein 4-Bit-Datenbus verwendet. Da der Baustein in CMOS-Technologie aufgebaut ist, verbraucht er sehr wenig Strom und läßt sich leicht mit einem Akku puffern. Einbrüche der Versorgungsspannung werden automatisch erkannt und mit einer Sperrung der Ein- und Ausgänge beantwortet. Der RTC ist mit einer Genauigkeit von 10 ppm erhältlich, also sehr präzise. Alle Funktionen sind in der *Tabelle* aufgeführt.
Die Funktionen 0...C dienen zum Lesen und Setzen von Tag, Datum und Uhrzeit. Bei der Angabe der Stunden-Zehner kann zwischen 12- und 24-Stunden-Ausgabe unterschieden werden. Mit Bit 2 und 3 der Tag-Zehner-Eingabe kann zwischen gregorianischem und dem japanischen Showa-Kalender unterschieden werden. Da für uns sicher nur der gregorianische Kalender wichtig ist, werden beide Bits auf Null gesetzt (das ist auch für die automatische Schaltjahr-Erkennung wichtig). Mit der Funktion D werden die Zeitstufen zurückgesetzt und die Funktionen E und F schalten die internen Zeitsignale auf die Datenleitungen D0...D3. Dann erscheint auf D0 ein Takt mit einer Frequenz von 1024 Hz, auf D1 wird jede Sekunde, auf D2 jede Minute und auf D3 jede Stunde ein Impuls ausgegeben. Damit kann man bei geeigneter Beschaltung eine Interrupt-Steuerung aufbauen.

Die Funktionen des RTC-58321

Funktion	Adr	Eingang A3	A2	A1	A0	Ausgang D3...D0	Wert	Bemerkung
Sek. Einer	0	L	L	L	L	aktueller Wert	0...9	
Sek. Zehner	1	L	L	L	H	aktueller Wert	0...5	
Min. Einer	2	L	L	H	L	aktueller Wert	0...9	
Min. Zehner	3	L	L	H	H	aktueller Wert	0...5	
Std. Einer	4	L	H	L	L	aktueller Wert	0...9	
Std. Zehner	5	L	H	L	H	aktueller Wert	0...1	D3 = H für 24 Std.-Anzeige D3 = L für 12 Std.-Anzeige
							0...2	D2 = H für P.M. D2 = L für A.M.
Wochentag	6	L	H	H	L	aktueller Wert	0...6	hier 0 = Mittwoch
Tag Einer	7	L	H	H	H	aktueller Wert	0...9	
Tag Zehner	8	H	L	L	L	aktueller Wert	0...3	D3/D2 für Schaltjahr
Monat Einer	9	H	L	L	H	aktueller Wert	0...9	
Monat Zehner	A	H	L	H	L	aktueller Wert	0...1	
Jahr Einer	B	H	L	H	H	aktueller Wert	0...9	
Jahr Zehner	C	H	H	L	L	aktueller Wert	0...9	
Reset	D	H	H	L	H			alle Zähler löschen
Test	E	H	H	H	L	Referenz-Sign.	0...F	Signale auf D0...D3
Test	F	H	H	H	H	Referenz-Sign.	0...F	Signale auf D0...D3

Die Schaltung

Wie bereits erwähnt, wird der RTC mit auf die Centronics-Schnittstelle gesetzt. Um hier zwischen Drucker- und RTC-Betrieb unterscheiden zu können, wird noch ein Tristate-Treiber benötigt. Er sorgt für die Entkopplung des RTC bei Druckerbetrieb. Die Freigabe des RTC erfolgt dann über Bit 6 des Ausgabe-Ports $41; ist dieses Bit Null, so kann auf den RTC zugegriffen werden. Die Daten werden über Bit 0...3 der Ein-/Ausgabe-Adresse $40 übertragen. Vier Widerstände sorgen für die TTL-Pegel auf diesen Leitungen. Zusätzlich ist eine Steuerlogik zur hardwaremäßigen Freigabe des

$\overline{\text{BUSY}}$-Signals bzw. des $\overline{\text{1MIN}}$-Impulses als Interrupt-Quelle vorhanden. Die Schaltung kann mit einer Batterie (1,5 V) gepuffert werden, die Dioden D1 und D2 sind für die Entkopplung vorgesehen. Die Steuerung des RTC erfolgt über die Ausgabe-Adresse $41 (ADRESSWRITE, WRITE, READ und CS2) sowie über die Eingabe-Adresse $41 ($\overline{\text{BUSY}}$). Da zur Verbindung zwischen RTC und Bus nur vier Datenleitungen vorhanden sind, werden diese im Multiplex-Verfahren benutzt. Die dabei verwendeten Steuerleitungen haben folgende Bedeutung:

Adr.-Write
Diese Leitung dient zur Adressierung der 16 internen Register des RTC. High-Pegel auf dieser Leitung teilt dem Baustein mit, daß sich auf dem Datenbus ein gültiges Adreßwort befindet. Der interne Adreßdecoder selektiert nun das entsprechende Register, das beim nächsten Zugriff als Ziel oder Quelle angesprochen werden kann.

Write
Die Steuerleitung für den Datentransfer zum RTC: Steht hier High-Pegel, so werden die Daten vom Bus in das vorher mit Adr.-Write selektierte Register übernommen.

Read
Die Steuerleitung für den Datentransfer vom RTC: High auf dieser Leitung bewirkt, daß der Inhalt des mit Adr.-Write ausgewählten Registers auf dem Datenbus ausgegeben wird.

CS2
Dies ist eine der beiden Baustein-Auswahl-Leitungen. Sind beide Leitungen aktiv (high), so ist der Baustein adressiert, andernfalls werden alle Ein-/Ausgabeleitungen des RTC inaktiv. In der Schaltung ist CS1 mit der Versorgungsspannung verbunden, womit bei fehlender Spannung jeder Zugriff gesperrt ist.

$\overline{\text{BUSY}}$
Mit diesem Ausgang signalisiert der Baustein, daß er nicht zur Datenübertragung bereit ist. Das ist immer dann der Fall, wenn der interne Zeitzähler um 1 Sekunde erhöht wird. Der Ausgang liefert also jede Sekunde einen negativen Impuls.

Der RTC hat noch zwei weitere Steuerleitungen, nämlich Test und Stop. Wie der Name schon sagt, dient der eine zu Testzwecken; eine 1 am Stop-Eingang hält den internen Zeitzähler an. In der vorliegenden Schaltung werden die beiden Eingänge nicht benutzt und sind daher an Masse gelegt.

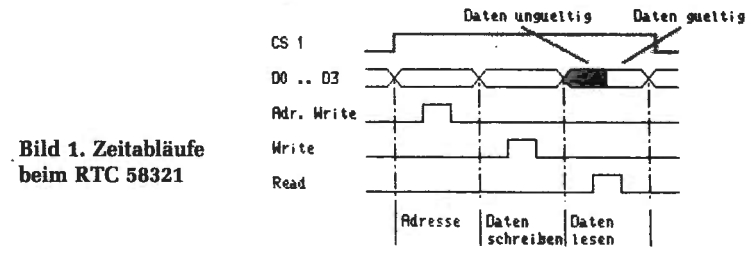


Bild 1. Zeitabläufe beim RTC 58321

Über die Ausgabe-Adresse $41 wird außerdem der Drucker (Bit 0) und die Interrupt-Freigabe (Bit 6, 7) gesteuert. Der Schreibschutz-Schalter verhindert ein unbeabsichtigtes Verstellen der Uhr, nur mit geöffnetem Schalter kann in den RTC geschrieben werden. Dies ist jedoch keine Einschränkung, da man die Zeit ja nur beim Wechsel zwischen Sommer- und Normalzeit sowie nach einem Batteriewechsel stellen muß.

Datentransfer

Ein Zugriff auf den Uhrenbaustein hat also den folgenden Ablauf:
- CS2 auf Eins legen
- Register-Adresse anlegen
- Adr.-Write auf Eins legen
- ADR.-Write wieder auf Null
- Schreibzugriff:
  - Daten anlegen
  - Write auf Eins legen
  - mindestens 2 µs warten
  - Write wieder auf Null
- Lesezugriff:
  - Read auf Eins legen
  - mindestens 2 µs warten
  - Daten auslesen
  - Read wieder auf Null
- CS2 wieder auf Null legen.

Ist der Baustein nicht in der Lage, einen Datentransfer durchzuführen, so wird dies durch eine Null am $\overline{\text{BUSY}}$-Ausgang angezeigt. Da dies immer dann auftritt, wenn der RTC seinen internen Zeitzähler um eine Sekunde erhöht, erscheint

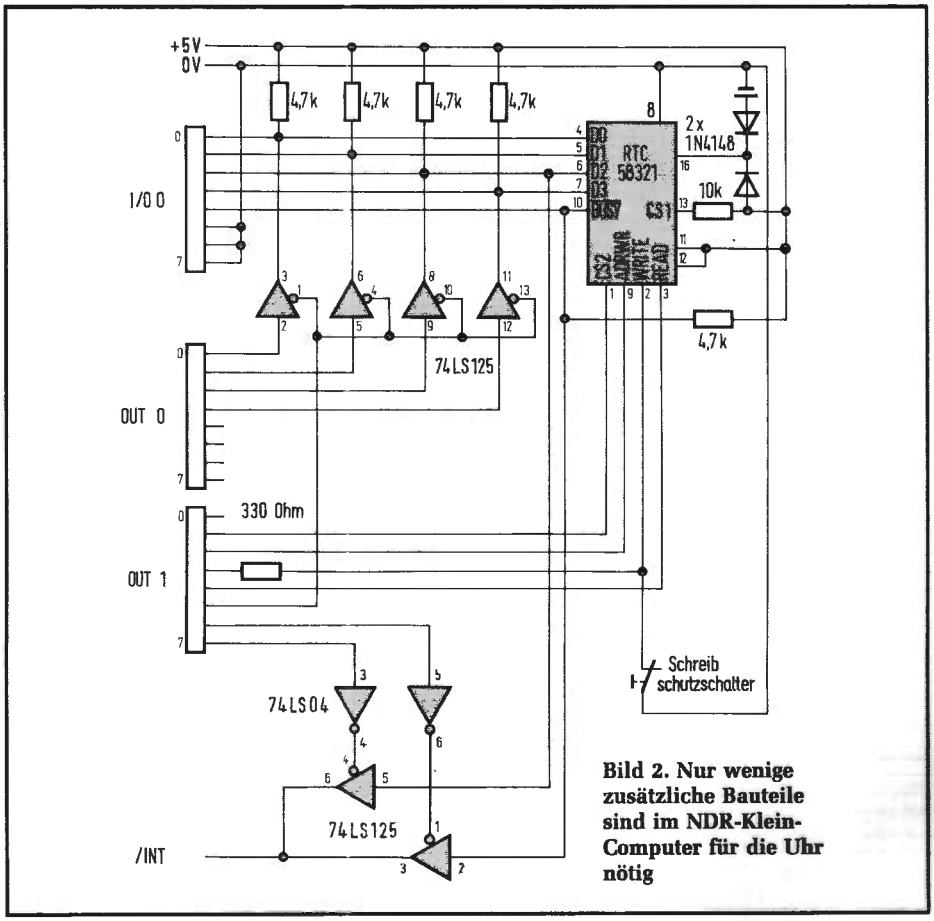


Bild 2. Nur wenige zusätzliche Bauteile sind im NDR-Klein-Computer für die Uhr nötig

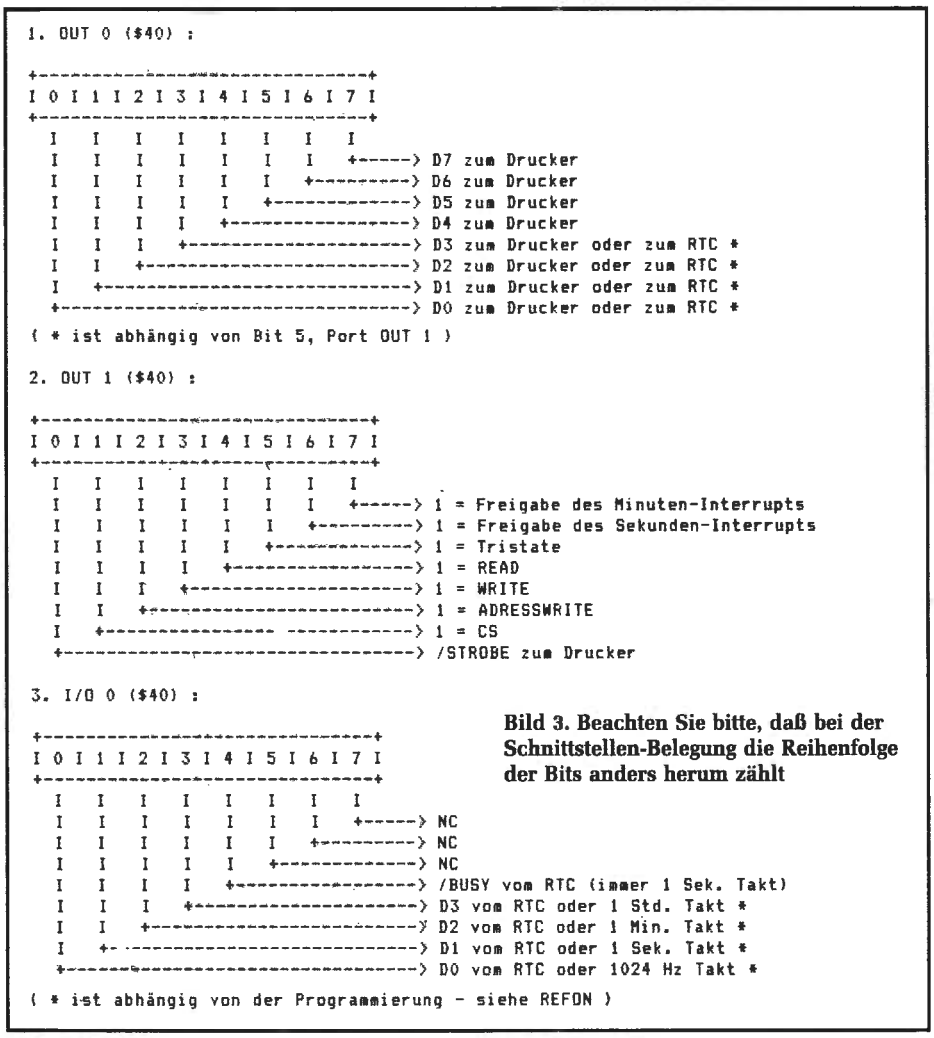


Bild 3. Beachten Sie bitte, daß bei der Schnittstellen-Belegung die Reihenfolge der Bits anders herum zählt

auch der $\overline{\text{BUSY}}$-Impuls genau im 1-Sekunden-Takt. Man kann damit einen Interrupt auslösen und die Uhr somit im Hintergrund laufen lassen. Ein Zeitdiagramm dazu zeigt *Bild 1*. Die Schaltung (*Bild 2*) läßt sich in Fädeltechnik realisieren. Die dadurch geschaffene Belegung der Ein- und Ausgabe-Adressen zeigt *Bild 3*.

Die Software

Das Treiberprogramm (*Bild 4*), mit dem der 68008 die Uhrenkarte bedienen kann, besteht im Wesentlichen aus zwei Teilen, auf denen alle anderen Routinen aufbauen: Dem Block mit den Unterprogrammen zur Programmierung der Handshake-Leitungen (AWRITE usw.), sozusagen die unterste Software-Schnittstelle zum RTC, und dem Teil, in dem diese Grundroutinen zum kompletten Lese- und Schreibzugriff auf den RTC zusammengefügt werden. Dabei wird jeweils in D0 die Adresse übergeben, während in D1 die Daten vom bzw. für den RTC stehen.

Aufbauend auf diese Routinen sind folgende Programme und Funktionen vorhanden:

RTCSHOW
An einer festgelegten Position auf dem Bildschirm werden Wochentag, Datum und Uhrzeit ausgegeben.

LESEN
Wochentag, Datum und Uhrzeit werden entweder mit wählbarer Größe und Position auf dem Bildschirm oder in einen Puffer ausgegeben. Das aufrufende Programm muß dazu folgende Parameter übergeben:

D0.B Schriftgröße
D1.W X-Position
D2.W Y-Position
D3.B Ausgabe (0 = Puffer, 1 = Bild)
D6.B Interrupt-Maske
A0.L Puffer-Adresse (23 Byte)

Bei Puffer-Ausgabe werden die Werte in D0 und D1 nicht beachtet.

SETUHR
Ein Menüprogramm zum Stellen der Uhr (*Bild 5*). Die Eingabe von Wochentag, Datum und Zeit erfolgt im ASCII-Format, Fehler bei der Eingabe werden gemeldet. Dabei ist zu beachten, daß zum Zeitpunkt der Übernahme (also beim Drücken der Return-Taste) der Schreibschutz-Schalter geöffnet sein muß.

HEXREAD
Wochentag, Datum und Uhrzeit werden hexadezimal in einen Puffer geschrieben. Beim Aufruf wird in A0 die Puffer-Adresse übergeben.

TRAPHANDLER
Mit diesem Programm können alle bisher genannten Funktionen über den Trap-#7-Befehl aufgerufen werden. Im RDK-Grundprogramm-RAM ist nämlich in einer Tabelle ein Sprung eingetragen, der auf ein eigenes Programm umgelenkt werden kann. Damit kann das Uhr-Programm von jedem anderen Programm benutzt werden. Wie bei allen Grundprogramm-Traps wird einfach im Register D7.W die Nummer der gewünschten Funktion eingetragen und anschließend mit Trap #7 die RTC-Routine aufgerufen. Die einzelnen Funktions-Codes und die Aufruf-Parameter sind im Bild 4 zu finden. Natürlich muß vor dem ersten Aufruf der Trap-Handler mit TRAPINIT installiert werden.
Damit auch andere Programme TRAPINIT finden und ausführen können, ist ein HEADER vorgesehen, nach dem man suchen und dann TRAPINIT ausführen kann. Der Trap-Vektor wird ebenfalls auf die Vektor-Adresse der Prozessors 68008 geschrieben. Liegt das Grundprogramm nicht ab Adresse $0, so wird damit der Umweg über die RAM-Tabelle gespart.

RTCINIT
Durch Aufrufen dieses Programmes wird der interrupt-gesteuerte Hintergrundbetrieb mit Ausgabe auf dem Bildschirm initialisiert. Zum Auslösen des Interrupts wird hier der $\overline{\text{BUSY}}$-Ausgang benutzt, wodurch die Zeitausgabe jede Sekunde erfolgt. Die Schaltung ist so ausgelegt, daß der Interrupt hard- und softwaremäßig freigegeben werden muß. Auch für diese Initialisierung ist wieder ein Programm mit Header und eine Demo-Routine vorhanden. Nach Ausführen von INTINIT erscheint die Uhrzeit in der linken unteren Ecke, bis ein Reset ausgelöst oder die Interrupt-Priorität von einem anderen Programm erhöht wird.

```
0C7000                  ORG $C7000      * ADRESSE DER ROUTINE IM EPROM
0C7000
0C7000                  OFFSET -$20000  * ABLAGEADRESSE RAM !!
0C7000                  KOPF:
0C7000  55AA0180        DC.L $55AA0180
0C7004  5254432D494E49  DC.B 'RTC-INIT'
0C700B  54
0C700C  0000046C        DC.L TRAPINIT-KOPF      * USERTRAP WIRD INITIALISIERT
0C7010  0001DD34        DC.L ENDEA-KOPF
0C7014  01 00 00 00 00  DC.B 1,0,0,0,0
0C7019  00              DS 0
0C701A
0C701A                  ***************
0C701A                  ******* ALLGEMEINE UNTERRDUTINEN ZUM BETRIEB DES RTC (RTC-TOOL)
0C701A                  ***************
0C701A
0C701A                  *** VEREINBARUNGEN
0C701A
= FFFFFF40              DIN     EQU $FFFFFF40           * PORT FUER DATENEINGANG VOM RTC
= FFFFFF40              DOUT    EQU $FFFFFF40           * DTO. FUER DATENAUSGANG
= FFFFFF41              CONTR   EQU $FFFFFF41           * ADRESSE DES CONTROLLPORTS VOM RTC
= FFFFFF68              TASTE   EQU $FFFFFF68           * TASTATURPORT
= 000000C7              AWRITES EQU  %11000111          * BINAERSELECTION FUER ADRESSWRITE
= 000000CB              WRITES  EQU  %11001011          * DTO. WRITE
= 000000F3              READS   EQU  %11110011          * DTO. READ
= 000000E1              TRISTS  EQU  %11100001          * DTO. TRI-STATE
= 0000003F              INTDIS  EQU  %00111111          * DTO. INTERRUPT ABSCHALTEN
= 000000BF              INTMIN  EQU  %10111111          * DTO. INTERRUPT JEDE MIN. HARDWAREMAESIG FREIG.
= 0000007F              INTSEK  EQU  %01111111          * DTO. INTERRUPT JEDE SEK. HARDWAREMAESIG FREIG.
0C701A
0C701A                  *       BIT:  ^^^^^^^^          BELEGUNG DES CONTROLLPORTS :
0C701A                  *       0     ^^^^^^^+--------  STROBE ZUM DRUCKER
0C701A                  *       1     ^^^^^^+---------  CS
0C701A                  *       2     ^^^^^+----------  ADRESSWRITE
0C701A                  *       3     ^^^^+-----------  WRITE
0C701A                  *       4     ^^^+------------  READ
0C701A                  *       5     ^^+-------------  TRI-STATE
0C701A                  *       6     ^+--------------  FREIGABE SEKUNDEN INTERRUPT
0C701A                  *       7     +---------------  FREIGABE MINUTEN INTERRUPT
0C701A
0C701A                  *** DIVERSE UNTERPROGRAMME
0C701A
0C701A                  BUSYTEST:                       * TEST OB RTC BEREIT
0C701A  1239 FFFFFF40   MOVE.B DIN,D1                   * DATEN STEHEN AUCH FUER ANDERE UPRG. ZUR VERF.
0C7020  0B01 0004       BTST.B #$4,D1                   * BUSY ?
0C7024  67F4            BEQ.S BUSYTEST                  * JA, DANN WARTEN
0C7026  4E75            RTS
0C7028
0C7028                  RTCWAIT:                        * BUSYTEST REICHT
0C7028  6100 FFF0       BSR BUSYTEST                    * ALS WAIT
0C702C  1A3C 00E1       MOVE.B #TRISTS,D5
0C7030  CA06            AND.B D6,D5                     * IN D6 STEHT CODE FUER INTERRUPTBETRIEB
0C7032  13C5 FFFFFF41   MOVE.B D5,CONTR                 * TRI-STATE-SELECT
0C7038  4E75            RTS
0C703A
0C703*                  READ:                           * DATEN VOM RTC LESEN
0C703A  1A3C 00F3       MOVE.B #READS,D5                * READ-SELECT
0C703E  CA06            AND.B D6,D5                     * IN D6 STEHT CODE FUER INTERRUPTBETRIEB
```

Bild 4. Nicht beliebig eingebunden, sondern voll kompatibel zur NDR-Uhrenkarte wird die Erweiterung durch dieses Programm

```
0C7040  13C5 FFFFFF41   MOVE.B D5,CONTR
0C7046  6100 FFE0       BSR RTCWAIT                     * ERGEBNISS = IN D1 (WURDE IN BUSYTEST GEHOLT !)
0C704A  4E75            RTS
0C704C
0C704C                  AWRITE:                         * ADRESSE AN RTC GEBEN
0C704C  1A3C 00C7       MOVE.B #AWRITES,D5              * AWRITE-SELECT
0E7050  CA06            AND.B D6,D5                     * IN D6 STEHT CODE FUER INTERRUPTBETRIEB
0C7052  13C5 FFFFFF41   MOVE.B D5,CONTR
0C7058  6100 FFCE       BSR RTCWAIT                     * WARTEN BIS RTC BEREIT
0C705C  4E75            RTS
0C705E
0C705E                  WRITE:                          * DATEN IN RTC SCHREIBEN
0C705E  1A3C 00CB       MOVE.B #WRITES,D5               * WRITE-SELECT
0C7062  CA06            AND.B D6,D5                     * IN D6 STEHT CODE FUER INTERRUPTBETRIEB
0C7064  13C5 FFFFFF41   MOVE.B D5,CONTR
0C706A  6100 FFBC       BSR RTCWAIT                     * WARTEN BIS RTC BEREIT
0C706E  4E75            RTS
0C7070
0C7070                  *** DATEN VOM RTC LESEN
0C7070
0C7070                  UPREAD:                 * HIER WIRD EIN WERT AUS DEM UHRENBAUSTEIN GELESEN
0C7070                                          * D0 = GEW. ADRESSE, D1 = ERGEBNIS IN HEX
0C7070  13C0 FFFFFF40   MOVE.B D0,DOUT          * ADRESSE ANLEGEN
0C7076  6100 FFD4       BSR AWRITE              * UND IN RTC SCHREIBEN
0C707A  6100 FFBE       BSR READ                * DANN DATEN AUSLESEN, ERGEBNISS IN D1
0C707E  4E75            RTS
0C7080
0C7080                  *** DATEN IN DEN RTC SCHREIBEN
0C7080
0C7080                  UPWRITE:                * HIER WIRD EIN WERT IN DEN UHRENBAUSTEIN GESCHRIEBEN
0C7080                                          * D0 = GEW. ADRESSE, D1 = ZU SCHREIBENDER WERT
0C7080  1401            MOVE.B D1,D2            * WERT RETTEN
0C7082  13C0 FFFFFF40   MOVE.B D0,DOUT          * ADRESSE ANLEGEN
0C7088  6100 FFC2       BSR AWRITE              * UND IN RTC SCHREIBEN
0C708C  13C2 FFFFFF40   MOVE.B D2,DOUT          * DATEN ANLEGEN
0C7092  6100 FFCA       BSR WRITE               * UND IN RTC SCHREIBEN
0C7096  4E75            RTS
0C7098
0C7098                  ***************
0C7098                  ******* NACHFOLGEND PROGRAMME ZUM BETRIEB DES RTC ALS UHR
0C7098                  ***************
0C7098
0C7098                  *** EIN-/AUSGABEBUFFER - WICHTIG, MUSS RAM SEIN ! (V 1.4 NUN IM GPRG.-BUFFER)
0C7098
= 00000FCA              OUT     EQU $1000-27*2          * ZWISCHENSPEICHER, LIEGT AN DER OBEREN GRENZE
= 00000FE4              OUT1    EQU OUT+26              * DER SYMBOLTABELLE !!
0C7098
0C7098                  *** UNTERPROGRAMM ZUR TEXTAUSGABE
0C7098
0C7098                  WRITECRT:                       * AUFRUF GRUNDPROGRAMMROUTINE WRITE
0C7098  3E3C 000A       MOVE #!WRITE,D7
0C709C  4E41            TRAP #1
0C709E  4E75            RTS
0C70A0
0C70A0                  CLR:                            * AUFRUF GRUNDPROGRAMMROUTINE CLR
0C70A0  1E3C 0010       MOVE.B #!CLR,D7
0C70A4  4E41            TRAP #1
0C70A6  4E75            RTS
0C70A8
0C70A8                  *** UHR STELLEN
0C70A8
```

```
0C70A8                  SETUHR:
0C70A8  6100 FFF6       BSR CLR                 * TEXTAUSGABE VORBEREITEN
0C70AC
0C70AC                  SETUHR1:
0C70AC  303C 0022       MOVE #$22,D0            * PARAMETER FUER TEXTGROESSE ETC. VORBELEGEN
0C70B0  343C 00C8       MOVE #200,D2
0C70B4  323C 000A       MOVE #10,D1
0C70B8  41FA 02C8       LEA TEXT1(PC),A0
0C70BC  6100 FFDA       BSR WRITECRT
0C70C0
0C70C0  0442 0014       SUB #20,D2
0C70C4  41FA 02C7       LEA TEXT2(PC),A0
0C70C8  6100 FFCE       BSR WRITECRT
0C70CC
0C70CC  41ED 0FCA       LEA OUT(A5),A0          * ADRESSE DES ABLAGEBUFFERS
0C70D0  343C 0064       MOVE #100,D2            * KOORDINATEN ETC.
0C70D4  323C 000A       MOVE #10,D1
0C70D8  303C 0033       MOVE #$33,D0
0C70DC  363C 0015       MOVE #21,D3
0C70E0  3E3C 000B       MOVE #!READ,D7          * DANN EINLESEN
0C70E4  4E41            TRAP #1
0C70E6
0C70E6  41ED 0FCA       LEA OUT(A5),A0          * ABLAGEBUFFER IN A0, BZW A1
0C70EA  43ED 0FE4       LEA OUT1(A5),A1
0C70EE  302B 0000       MOVE.W $0(A0),D0        * WOCHENTAG HOLEN
0C70F2  0C40 4D6F       CMP.W #'Mo',D0          * MO ?
0C70F6  6700 005A       BEQ SMO                 * JA, DANN MONTAG SETZEN USW.
0C70FA  0C40 4469       CMP.W #'Di',D0
0C70FE  6700 005A       BEQ SDI
0C7102  0C40 4D69       CMP.W #'Mi',D0
0C7106  6700 005A       BEQ SMI
0C710A  0C40 446F       CMP.W #'Do',D0
0C710E  6700 005A       BEQ SDO
0C7112  0C40 4672       CMP.W #'Fr',D0
0C7116  6700 005A       BEQ SFR
0C711A  0C40 5361       CMP.W #'Sa',D0
0C711E  6700 005A       BEQ SSA
0C7122  0C40 536F       CMP.W #'So',D0
0C7126  6700 005A       BEQ SSO
0C712A
0C712A  103C 0022       MOVE.B #$22,D0          * WENN FEHLER, Z. B. STATT 'Mo' = 'mo'
0C712E  123C 000A       MOVE.B #10,D1
0C7132  143C 001E       MOVE.B #30,D2
0C7136  41FA 0278       LEA TEXT3(PC),A0        * FEHLERMELDUNG
0C713A  6100 FF5C       BSR WRITECRT
0C713E  123C 000A       MOVE.B #10,D1
0C7142  143C 000A       MOVE.B #10,D2
0C7146  41FA 0278       LEA TEXT4(PC),A0        * FEHLERMELDUNG
0C714A  6100 FF4C       BSR WRITECRT
0C714E
0C714E  6000 FF5C       BRA SETUHR1
0C7152
0C7152                  SMO:                    * WERTE FUER ENTSPRECHENDEN WOCHENTAG IN ABLAGEBUFFER
0C7152  103C 0005       MOVE.B #5,D0            * SCHREIBEN, HIER MONTAG, ANDERE DTO.
0C7156  6000 002E       BRA SETWO
0C715A
0C715A                  SDI:
0C715A  103C 0006       MOVE.B #6,D0
0C715E  6000 0026       BRA SETWO
0C7162
0C7162                  SMI:
0C7162  103C 0000       MOVE.B #0,D0
0C7166  6000 001E       BRA SETWO
0C716A
0C716A                  SDO:
0C716A  103C 0001       MOVE.B #1,D0
0C716E  6000 0016       BRA SETWO
0C7172
0C7172                  SFR:
0C7172  103C 0002       MOVE.B #2,D0
0C7176  6000 000E       BRA SETWO
0C717A
0C717A                  SSA:
0C717A  103C 0003       MOVE.B #3,D0
0C717E  6000 0006       BRA SETWO
0C7182
0C7182                  SSO:
0C7182  103C 0004       MOVE.B #4,D0
0C7186
0C7186                  SETWO:
0C7186  1340 0006       MOVE.B D0,$6(A1)        * HEXWERT FUER WOCHENTAG SETZEN
0C718A
0C718A  102B 0003       MOVE.B $3(A0),D0        * HIER DANN RESTLICHE DATEN SETZEN
0C718E  0440 0030       SUB #$30,D0             * VON ASCII IN HEX
0C7192  08C0 0003       BSET #3,D0              * FUER 24 STD. UHR
0C7196  1340 0005       MOVE.B D0,$5(A1)        * STD. ZEHNER
0C719A
0C719A  102B 0004       MOVE.B $4(A0),D0
0C719E  0440 0030       SUB #$30,D0
0C71A2  1340 0004       MOVE.B D0,$4(A1)        * STD. EINER
0C71A6
0C71A6  102B 0006       MOVE.B $6(A0),D0
0C71AA  0440 0030       SUB #$30,D0
0C71AE  1340 0003       MOVE.B D0,$3(A1)        * MIN. ZEHNER
0C71B2
0C71B2  102B 0007       MOVE.B $7(A0),D0
0C71B6  0440 0030       SUB #$30,D0
0C71BA  1340 0002       MOVE.B D0,$2(A1)        * MIN. EINER
0C71BE
0C71BE  102B 0009       MOVE.B $9(A0),D0
0C71C2  0440 0030       SUB #$30,D0
0C71C6  1340 0001       MOVE.B D0,$1(A1)        * SEK. ZEHNER
0C71CA
0C71CA  102B 000A       MOVE.B $A(A0),D0
0C71CE  0440 0030       SUB #$30,D0
0C71D2  1340 0000       MOVE.B D0,$0(A1)        * SEK. EINER
0C71D6
0C71D6  102B 000C       MOVE.B $C(A0),D0
0C71DA  0440 0030       SUB #$30,D0
0C71DE  1340 0008       MOVE.B D0,$8(A1)        * TAG ZEHNER
0C71E2
0C71E2  102B 000D       MOVE.B $D(A0),D0
0C71E6  0440 0030       SUB #$30,D0
0C71EA  1340 0007       MOVE.B D0,$7(A1)        * TAG EINER
0C71EE
0C71EE  102B 000F       MOVE.B $F(A0),D0
0C71F2  0440 0030       SUB #$30,D0
0C71F6  1340 000A       MOVE.B D0,$A(A1)        * MONAT ZEHNER
0C71FA
0C71FA  102B 0010       MOVE.B $10(A0),D0
0C71FE  0440 0030       SUB #$30,D0
0C7202  1340 0009       MOVE.B D0,$9(A1)        * MONAT EINER
```

```
0C7206
0C7206  1028 0012       MOVE.B $12(A0),D0
0C720A  0440 0030       SUB #$30,D0
0C720E  1340 000C       MOVE.B D0,$C(A1)            * JAHR ZEHNER
0C7212
0C7212  1028 0013       MOVE.B $13(A0),D0
0C7216  0440 0030       SUB #$30,D0
0C721A  1340 000B       MOVE.B D0,$B(A1)            * JAHR EINER
0C721E
0C721E  41ED 0FE4       LEA OUT1(A5),A0             * ABLAGEADRESSE DES STRINGS IN A0
0C7222  4280            CLR.L D0                    * ADRESSPOINTER LOESCHEN
0C7224  4281            CLR.L D1
0C7226
0C7226                  SETLOOP:
0C7226  1218            MOVE.B (A0)+,D1             * AUS ABLAGE IN D1
0C7228  6100 FE56       BSR UPWRITE                 * AN UHR GEBEN
0C722C  5200            ADDQ.B #1,D0                * ADRESSPOINTER ERHOEHEN
0C722E  0C00 000C       CMP.B #$C,D0                * ALLES UEBERTRAGEN ?
0C7232  6E00 0006       BGT SETEND                  * JA, DANN ENDE
0C7236  6000 FFEE       BRA SETLOOP                 * SONST WEITER
0C723A
0C723A                  SETEND:
0C723A  4E75            RTS
0C723C
0C723C                  *** ANZEIGE VON WOCHENTAG, ZEIT UND DATUM AUF BILDSCHIRM ODER IN PUFFER
0C723C
0C723C                  RTCSHOW:                    * ANZEIGE AUF BILDSCHIRM, DIREKT-EINSPRUNG
0C723C  363C 0001       MOVE #1,D3                  * AUFRUF. DAZU CODE IN D3
0C7240  41ED 0FCA       LEA OUT(A5),A0              * UND PUFFERADRESSE IN A0
0C7244  203C 00000011   MOVE.L #$11,D0              * KOORDINATEN BELEGEN
0C724A  243C 000000F7   MOVE.L #247,D2
0C7250  223C 0000017C   MOVE.L #380,D1
0C7256  383C 0001       MOVE #1,D4                  * SEITENZAEHLER VORBELEGEN
0C725A  1C3C 003F       MOVE.B #INTDIS,D6           * CODE FUER INTERRUPTBETRIEB (HIER INT-DISABLE)
0C725E
0C725E                  LESEN:
0C725E  48E7 E000       MOVEM.L D0-D2,-(A7)         * KOORDINATEN RETTEN, BELEGUNG WIE BEI @WRITE
0C7262  48E7 1000       MOVEM.L D3,-(A7)            * D3 = 1 WENN ANZEIGE, D3 = 0 WENN NUR IN PUFFER
0C7266  6100 0014       BSR LESELP                  * IN A0 STEHT ADRESSE EINE PUFFERSPEICHERS
0C726A  4CDF 0008       MOVEM.L (A7)+,D3            * D3 HOLEN, IN D3 STEHT CODE,
0C726E  4CDF 0007       MOVEM.L (A7)+,D0-D2         * KOORDINATEN ZURUECKHOLEN
0C7272  0C03 0001       CMP.B #1,D3                 * PRUEFUNG, OB ANZEIGE ODER AUSGABE IN PUFFER
0C7276  6700 00D0       BEQ SHOW                    * 0 = AUSGABE IN PUFFER, 1 = BILDSCHIRM
0C727A  4E75            RTS                         * RUECKSPRUNG, WENN KEINE AUSGABE
0C727C
0C727C                  LESELP:                     * LEGT DATEN VON RTC ALS ASCII IN PUFFER
0C727C  48E7 00B0       MOVEM.L A0,-(A7)            * ADRESSE DES PUFFERS AUF STACK RETTEN
0C7280  4280            CLR.L D0                    * ADRESSZAEHLER
0C7282  45FA 00F0       LEA TABTAB(PC),A2           * ADRESSE DER WOCHENTAGSTABELLE
0C7286  303C 0006       MOVE #6,D0                  * DANN ENTSPRECHENDEN WERT IN AUSGABEBUFFER
0C728A  4281            CLR.L D1
0C728C  6100 FDE2       BSR UPREAD
0C7290  0201 000F       ANDI.B #$F,D1
0C7294  C2FC 0002       MULU #2,D1
0C7298  4CB2 0002 1000  MOVEM.W $0(A2,D1.W),D1
0C729E  30C1            MOVE.W D1,(A0)+
0C72A0  10FC 0020       MOVE.B #' ',(A0)+
0C72A4  103C 0008       MOVE.B #8,D0                * DANN RESTLICHE DATEN HOLEN
0C72A8  6100 00B0       BSR GET
0C72AC  103C 0007       MOVE.B #7,D0
0C72B0  6100 0078       BSR GET
0C72B4  10FC 002E       MOVE.B #'.',(A0)+
0C72B8  103C 000A       MOVE.B #10,D0
0C72BC  6100 006C       BSR GET
0C72C0  103C 0009       MOVE.B #9,D0
0C72C4  6100 0064       BSR GET
0C72C8  10FC 002E       MOVE.B #'.',(A0)+
0C72CC  103C 000C       MOVE.B #12,D0
0C72D0  6100 005B       BSR GET
0C72D4  103C 000B       MOVE.B #11,D0
0C72D8  6100 0050       BSR GET
0C72DC  10FC 0020       MOVE.B #' ',(A0)+
0C72E0  10FC 002D       MOVE.B #'-',(A0)+
0C72E4  10FC 0020       MOVE.B #' ',(A0)+
0C72E8  103C 0005       MOVE.B #5,D0
0C72EC  6100 003C       BSR GET
0C72F0  103C 0004       MOVE.B #4,D0
0C72F4  6100 0034       BSR GET
0C72F8  10FC 003A       MOVE.B #':',(A0)+
0C72FC  103C 0003       MOVE.B #3,D0
0C7300  6100 0028       BSR GET
0C7304  103C 0002       MOVE.B #2,D0
0C7308  6100 0020       BSR GET
0C730C  10FC 003A       MOVE.B #':',(A0)+
0C7310  103C 0001       MOVE.B #1,D0
0C7314  6100 0014       BSR GET
0C7318  103C 0000       MOVE.B #0,D0
0C731C  6100 000C       BSR GET
0C7320  10FC 0000       MOVE.B #0,(A0)+
0C7324  4CDF 0100       MOVEM.L (A7)+,A0            * PUFFERADRESSE RETOUR
0C7328  4E75            RTS
0C732A
0C732A                  GET:                        * LIEST DIE WERTE VOM RTC
0C732A  4281            CLR.L D1
0C732C  6100 FD42       BSR UPREAD
0C7330  0201 000F       ANDI.B #$F,D1
0C7334  0C00 0005       CMPI.B #5,D0                * STD. ZEHNER ?
0C7338  6600 0006       BNE NH10                    * NEIN, DANN WEITER
0C733C  0201 0003       ANDI.B #$3,D1               * SONST BIT 3 ZURUECKSETZEN (24 STD. UHR)
0C7340
0C7340                  NH10:
0C7340  0001 0030       ORI.B #$30,D1               * IN ASCII WANDELN
0C7344  10C1            MOVE.B D1,(A0)+
0C7346  4E75            RTS
0C7348
0C7348                  SHOW:                       * AM BILDSCHIRM ANZEIGEN
0C7348  48E7 C000       MOVEM.L D0-D1,-(A7)         * AUF STACK RETTEN
0C734C  6100 FD4A       BSR WRITECRT
0C7350  1004            MOVE.B D4,D0                * MIT D4 WIRD DIE SCHREIB- UND
0C7352  4004            NEGX.B D4                   * LESESEITE FUER FLIMMERFREIE
0C7354  0204 0003       ANDI.B #3,D4                * GRAFIK UMGESCHALTET
0C7358  1204            MOVE.B D4,D1
0C735A  3E3C 001B       MOVE #!NEWPAGE,D7
0C735E  4E41            TRAP #1
0C7360  4CDF 0003       MOVEM.L (A7)+,D0-D1
0C7364  43F9 FFFFFF68   LEA TASTE,A1                * ENDE MIT CONTROL C
0C736A  0C11 0003       CMPI.B #$03,(A1)
0C736E  6600 FEEE       BNE LESEN
0C7372  4E75            RTS
0C7374
0C7374                  *** DATEN-TABELLEN
0C7374
```

```
0C7374                  TAGTAB:                         * WOCHENTAGSTABELLE ( 0 = MITTWOCH !)
0C7374  4D69446F467253  DC.B 'Mi','Do','Fr','Sa','So','Mo','Di'
0C737B  61536F4D6F4469
0C7382
0C7382                  *** ABLAGEBEREICH FUER TEXTE
0C7382
0C7382  556B7220736574  TEXT1:  DC.B 'Uhr setzen',0
0C7389  7A656E00
0C738D  446174656E666F  TEXT2:  DC.B 'Datenformat = Tg/St/Mi/Se/Da/Mo/Ja',0
0C7394  726D6174203D20
0C739B  54672F53742F4D
0C73A2  692F53652F4461
0C73A9  2F4D6F2F4A6100
0C73B0  45696E67616265  TEXT3:  DC.B 'Eingabefehler !',0
0C73B7  6665686C657220
0C73BE  2100
0C73C0  466F726D617420  TEXT4:  DC.B 'Format z. B. Mo/12/30/00/13/08/86',0
0C73C7  7A2E20422E204D
0C73CE  6F2F31322F3330
0C73D5  2F30302F31332F
0C73DC  30382F383600
0C73E2                  DS 0
0C73E2
0C73E2                  ++++++++++++++++
0C73E2                  +++++++ PROGRAMM ZUM AUFRUF DER RTC-FUNKTIONEN UEBER USERTRAB
0C73E2                  ++++++++++++++++
0C73E2
= 0000009C              TRAPJMPP EQU $9C                * TRAP 7 EXCEPTION-VECTORTABELLE 68000
= 00000A16              TRAPJMPB EQU $A16               * IM GRUNDPROGRAMM-SPEICHER
0C73E2
0C73E2                  *** USER-TRAP HANDLER
0C73E2
0C73E2                  * IMPLEMENTIERTE FUNKTIONEN (CODE IN D7.W) :
0C73E2                  * #700 = DATEN VON RTC LESEN (D0 GEW. ADRESSE, D1 DATEN RETOUR)
0C73E2                  * #701 = DATEN IN RTC SCHREIBEN (D0 GEW. ADRESSE, D1 DATEN FUER RTC)
0C73E2                  * #702 = UHR LESEN (D0 GROESSE, D1,D2 KOORDINATEN, A0 PUFFER, D4 CRT / PUFFER)
0C73E2                  * #703 = UHRZEITANZEIGE AUF BILDSCHIRM
0C73E2                  * #704 = UHR STELLEN (MENUEGESTEUERT)
0C73E2                  * #705 = RTC-DATEN IM HEX-FORMAT IN PUFFER LESEN (PUFFERADRESSE IN A0)
0C73E2                  * #706 = REFERENZSIGNALE AUF D0-D3
0C73E2                  * #707 = EMULATION @GETUHR
0C73E2                  * #708 = EMULATION @SETUHR
0C73E2
0C73E2                  USERTRAP:                       * NR. DER GEW. FUNKTION IN D7.W
0C73E2  0C47 02BC       CMP #700,D7                     * DATEN VOM RTC LESEN ?
0C73E6  6600 0008       BNE UT1
0C73EA  6100 FC84       BSR UPREAD
0C73EE  4E73            RTE
0C73F0
0C73F0                  UT1:
0C73F0  0C47 02BD       CMP #701,D7                     * DATEN IN RTC SCHREIBEN ?
0C73F4  6600 0008       BNE UT2
0C73F8  6100 FC86       BSR UPWRITE
0C73FC  4E73            RTE
0C73FE
0C73FE                  UT2:
0C73FE  0C47 02BE       CMP #702,D7                     * UHR LESEN (CRT / PUFFER)
0C7402  6600 0008       BNE UT3
0C7406  6100 FE56       BSR LESEN
0C740A  4E73            RTE
0C740C
```

```
0C740C                  UT3:
0C740C  0C47 02BF       CMP #703,D7                     * UHRZEIT ANZEIGEN (DAUERANZEIGE)
0C7410  6600 0008       BNE UT4
0C7414  6100 FE26       BSR RTCSHOW
0C7418  4E73            RTE
0C741A
0C741A                  UT4:
0C741A  0C47 02C0       CMP #704,D7
0C741E  6600 0008       BNE UT5
0C7422  6100 FC84       BSR SETUHR
0C7426  4E73            RTE
0C7428
0C7428                  UT5:
0C7428  0C47 02C1       CMP #705,D7
0C742C  6600 0008       BNE UT6
0C7430  6100 0064       BSR HEXREAD
0C7434  4E73            RTE
0C7436
0C7436                  UT6:
0C7436  0C47 02C2       CMP #706,D7
0C743A  6600 0008       BNE UT7
0C743E  6100 003C       BSR REFON
0C7442  4E73            RTE
0C7444
0C7444                  UT7:
0C7444  0C47 02C3       CMP #707,D7
0C7448  6600 0008       BNE UT8
0C744C  6100 0166       BSR GETUHR
0C7450  4E73            RTE
0C7452
0C7452                  UT8:
0C7452  0C47 02C4       CMP #708,D7
0C7456  6600 0008       BNE UT9
0C745A  6100 01B4       BSR SETUHR
0C745E  4E73            RTE
0C7460
0C7460                  UT9:
0C7460  3E3C FFFF       MOVE #$FFFF,D7                  * FEHLERCODE, WENN FUNKTION NICHT DA
0C7464  4E73            RTE
0C7466
0C7466                  *** INITIALISIEREN DES USER-TRAP
0C7466
0C7466                  DS 0
0C7466  55AA            DC.W $55AA                      * HEADER ZUR KENNUNG, DAMIT KOENNE ANDERE
0C7468  2211AA55        DC.L $2211AA55                  * PROGRAMME DIESE ROUTINE SUCHEN UND AUSFUEHREN
0C746C
0C746C                  TRAPINIT:                       * ADRESSE IN TRAP-VECTOR EINTRAGEN
0C746C  41FA FF74       LEA USERTRAP(PC),A0             * DAZU ADRESSE IN A0 HOLEN UND VON DORT
0C7470  23C8 0000009C   MOVE.L A0,TRAPJMPP             * IN ENTSPRECHENDE SPEICHER EINTRAGEN,
0C7476  2B48 0A16       MOVE.L A0,TRAPJMPB(A5)          * DADURCH RELOKATIV !
0C747A  4E75            RTS                             * BENUTZT WIRD TRAP #7
0C747C
0C747C                  ++++++++++++++++
0C747C                  +++++++ PROGRAMM FUER TIMER-OUTPUT AUF D0-D4
0C747C                  ++++++++++++++++
0C747C
0C747C                  *       D0 = 1024 HZ, D1 = /1 SEK., D2 = /1 MIN., D3 = /1 STD.
0C747C                  *       IMPULSE D0 = POSITIVE LOGIK - IMPULSLAENGE = 488,3 US ( 1 ZU 1 )
0C747C                  *       IMPULSE D1-D3 = NEGATIVE LOGIK - IMPULSLAENGE = 122,1 US
0C747C
0C747C                  REFON:                          * SCHALTET REFERENZIMPULSE AUF D0-D3
```

```
0C747C  103C 00FE       MOVE.B #$FE,D0              * CODE IN D0
0C7480  13C0 FFFFFF40   MOVE.B D0,DOUT              * DANN AUF DATENBUS
0C7486  6100 FBC4       BSR AWRITE                  * UND AN ADRESSLATCH IM RTC UEBERGEBEN
0C748A  103C 0033       MOVE.B #%00110011,D0        * CONDITIONS IN D0 (2.6 INT=DISABLE)
0C748E  13C0 FFFFFF41   MOVE.B D0,CONTR             * AUF CONTROLLPORT GEBEN
0C7494  4E75            RTS                         * NUN ERSCHEINEN IMPULSE AUF DEN DATENLEITUNGEN
0C7496
0C7496                  ***************
0C7496                  ******* PROGRAMM ZUM AUSLESEN DER RTC-DATEN IM HEX-FORMAT IN PUFFER A0 -> PUFFER
0C7496                  ***************
0C7496
0C7496                  HEXREAD:
0C7496  43FA 0030       LEA READTAB(PC),A1          * IN DER TABELLE STEHT REIHENFOLGE DER LESEDATEN
0C749A  1C3C 003F       MOVE.B #INTDIS,D6           * NICHT MIT INTERRUPT
0C749E
0C749E                  HEXRLP:                     * LIEST ALLE DATEN VOM RTC, DAZU DIE GEW. ADR
0C749E  1019            MOVE.B (A1)+,D0             * IN D0,
0C74A0  0C00 00FF       CMP.B #$FF,D0               * WENN $FF, DANN ENDE
0C74A4  6700 001C       BEQ HEXREND
0C74A8  6100 FBC6       BSR UPREAD                  * DER WERT WIRD DANN IN D1 ZURUECKGELIEFERT
0C74AC  0201 000F       ANDI.B #$F,D1               * BIT 4-7 RESET
0C74B0  0C00 0005       CMPI.B #5,D0                * STD. ZEHNER ?
0C74B4  6600 0006       BNE HEXRLP1                 * NEIN, DANN WEITER
0C74B8  0201 0003       ANDI.B #$3,D1               * SONST BIT 3 ZURUECKSETZEN (24 STD. UHR)
0C74BC
0C74BC                  HEXRLP1:
0C74BC  10C1            MOVE.B D1,(A0)+             * UND DANN IN PUFFER
0C74BE  6000 FFDE       BRA HEXRLP                  * UND WEITER
0C74C2
0C74C2                  HEXREND:
0C74C2  10BC 00FF       MOVE.B #$FF,(A0)            * ALS ENDEZEICHEN
0C74C6  4E75            RTS
0C74C8
0C74C8                  READTAB:                    * HIER STEHEN DIE LESEADRESSEN
0C74C8  06 08 07 0A 09  DC.B $6,$8,$7,$A,$9,$C,$B,$5,$4,$3,$2,$1,$0,$FF
0C74CD  0C 0B 05 04 03
0C74D2  02 01 00 FF
0C74D6
0C74D6                  *************
0C74D6                  ******* PROGRAMM ZUM BETRIEB ES RTC UEBER INTERRUPT
0C74D6                  *************
0C74D6
= 00000074              INT5JMPP EQU $74            * INT LVL. 5 VOM PROZESSOR
= 0000008B              INT5JMPG EQU $08            * DTD. VOM GRUNDPROGRAMM (UEBER (A5) ADRESSIERT)
0C74D6
0C74D6                  *** INITIALISIEREN DES INTERRUPTS
0C74D6
0C74D6  55AA            DC.W $55AA                  * HEADER ZUR KENNUNG, DAMIT KOENNE ANDERE
0C74D8  1122AA55        DC.L $1122AA55              * PROGRAMME DIESE ROUTINE SUCHEN UND AUSFUEHREN
0C74DC
0C74DC                  INTINIT:                    * ADRESSE IN INT-VECTOR EINTRAGEN
0C74DC  41FA 001A       LEA RTCINT(PC),A0           * DAZU ADRESSE IN A0 HOLEN UND VON DORT
0C74E0  23C8 00000074   MOVE.L A0,INT5JMPP          * IN ENTSPRECHENDE SPEICHER EINTRAGEN,
0C74E6  2B48 0008       MOVE.L A0,INT5JMPG(A5)      * DADURCH RELOKATIV !
0C74EA  46FC 2000       MOVE #$2000,SR              * DANN NOCH INTERRUPT FREIGEBEN
0C74EE  13FC 007F       MOVE.B #INTSEK,CONTR        * AUCH HARDWAREMAESIG !
0C74F2  FFFFFF41
0C74F6  4E75            RTS                         * BENUTZT WIRD INTLVL 5 = LTG. /INT
0C74F8
0C74F8                  *** INTERRUPTROUTINE
0C74F8
```

```
= FFFFFF70              GDP     EQU $FFFFFF70       * ADRESSE GDP
= 00000217              VIEWPAG EQU $217            * HIER STEHT DIE AKT. LESESEITE (REL. -> A5)
= 00000218              WRTPAGE EQU $218            * HIER STEHT DIE AKT. WRITEPAGE (REL. -> A5)
0C74F8
0C74F8                  RTCINT:                     * HIER WIRD TAG, DATUM UND UHRZEIT ANGEZEIGT
0C74F8  48E7 FFFF       MOVEM.L D0-D7/A0-A7,-(A7)   * ALLE REGISTER RETTEN
0C74FC  1C3C 007F       MOVE.B #INTSEK,D6           * INT HARDWAREMAESIG FREIGEBEN
0C7500  363C 0000       MOVE #0,D3                  * GEHT UEBER ROUTINE LESEN, ANZEIGE-CODE IN D3
0C7504  41ED 0FCA       LEA OUT(A5),A0              * UND PUFFERADRESSE IN A0
0C7508  6100 FD54       BSR LESEN                   * DANN IN PUFFER LESEN
0C750C  6100 0074       BSR GDPSAVE                 * GDP - REGISTER RETTEN
0C7510  48E7 0600       MOVEM.L D5-D6,-(A7)         * WERDEN BENOETIGT
0C7514  1A2D 0217       MOVE.B VIEWPAG(A5),D5       * UM SCHREIB- UND LESESEITE ZU RETTEN
0C7518  1C2D 0218       MOVE.B WRTPAGE(A5),D6
0C751C  103C 0000       MOVE.B #0,D0                * NUN AUSGABE AUF SEITE 0
0C7520  123C 0000       MOVE.B #0,D1
0C7524  3E3C 001B       MOVE #!NEWPAGE,D7
0C7528  4E41            TRAP #1
0C752A  41ED 0FCA       LEA OUT(A5),A0              * PUFFERADRESSE
0C752E  D1FC 00000003   ADDA.L #3,A0                * WOCHENTAG NICHT ANZEIGEN (PASST NICHT !)
0C7534  103C 0011       MOVE.B #$11,D0              * GROESSE
0C7538  323C 018B       MOVE #395,D1
0C753C  343C 0001       MOVE #1,D2
0C7540  6100 FB56       BSR WRITECRT                * AUSGABE AUF BILDSCHIRM
0C7544  103C 0000       MOVE.B #0,D0                * DANN NOCH AUF SEITE 1
0C7548  123C 0000       MOVE.B #0,D1
0C754C  3E3C 001B       MOVE #!NEWPAGE,D7
0C7550  4E41            TRAP #1
0C7552  41ED 0FCA       LEA OUT(A5),A0              * PUFFERADRESSE
0C7556  D1FC 00000003   ADDA.L #3,A0                * WOCHENTAG NICHT ANZEIGEN (PASST NICHT !)
0C755C  103C 0011       MOVE.B #$11,D0              * GROESSE
0C7560  323C 018B       MOVE #395,D1
0C7564  343C 0001       MOVE #1,D2
0C7568  6100 FB2E       BSR WRITECRT                * AUSGABE AUF BILDSCHIRM
0C756C  1B45 0217       MOVE.B D5,VIEWPAG(A5)       * DANN SCHREIB- UND LESESETE RETOUR
0C7570  1B46 0218       MOVE.B D6,WRTPAGE(A5)
0C7574  4CDF 0060       MOVEM.L(A7)+,D5-D6          * REGISTER DTO.
0C7578  6100 001E       BSR GDPLOAD                 * UND GDP - REGISTER RETOUR
0C757C  4CDF FFFF       MOVEM.L (A7)+,D0-D7/A0-A7   * REGISTER RETOUR
0C7580  4E73            RTE
0C7582
0C7582                  *** RETTEN DER GDP - REGISTER
0C7582
0C7582                  GDPSAVE:                    * RETTET DIE REGISTER IN EINEN PUFFER
0C7582  41F9 FFFFFF71   LEA GDP+1,A0                * ADRESSE GDP, NUR CMD NICHT RETTEN
0C7588  43ED 0FE4       LEA OUT1(A5),A1             * HIER REGISTER ABLEGEN
0C758C  323C 000A       MOVE #15-4-1,D1             * ANZAHL
0C7590
0C7590                  SAVELP:
0C7590  12D8            MOVE.B (A0)+,(A1)+
0C7592  51C9 FFFC       DBRA D1,SAVELP
0C7596  4E75            RTS                         * WENN ALLES ABGESPEICHERT
0C7598
0C7598                  *** ZURUECKSCHREIBEN DER GDP-REGISTER
0C7598
0C7598                  GDPLOAD:                    * GDP REGISTER ZURUECK
0C7598  41F9 FFFFFF71   LEA GDP+1,A0                * ADRESSE GDP, NUR CMD NICHT RETTEN
0C759E  43ED 0FE4       LEA OUT1(A5),A1             * HIER REGISTER ABLEGEN
0C75A2  323C 000A       MOVE #15-4-1,D1             * ANZAHL
0C75A6
0C75A6                  LOADLP:
```

```
0C75A6  3E3C 0012       MOVE #!WAIT,D7              * WARTEN BIS GDP READY
0C75AA  4E41            TRAP #1
0C75AC  10D9            MOVE.B (A1)+,(A0)+
0C75AE  51C9 FFF6       DBRA D1,LOADLP
0C75B2  4E75            RTS                         * WENN ALLES ZURUECKGESCHRIEBEN
0C75B4
0C75B4                  ***************
0C75B4                  ******* HIER PROGRAMME ZUR NACHBILDUNG DER ORIGINAL-UHR MIT E05-16
0C75B4                  ***************
0C75B4
0C75B4                  *** ROUTINE LEGT DIE DATEN WIE DIE GRUNDPROGRAMMROUTINE @GETUHR IM BCD-FORMAT
0C75B4                  *   IN DURCH A0 ADRESSIERTEN PUFFER AB
0C75B4
0C75B4                  EGETUHR:
0C75B4  48E7 FFC0       MOVEM.L A0-A1/D0-D7,-(A7)   * REGISTER RETTEN
0C75B8  43FA 00BB       LEA EMUTAB(PC),A1           * LESETABELLE IN A0
0C75BC  42B0            CLR.L D0
0C75BE  1C3C 003F       MOVE.B #INTDIS,D6
0C75C2
0C75C2                  EGETLP:
0C75C2  4281            CLR.L D1
0C75C4  1019            MOVE.B (A1)+,D0             * ADRESSE IN D0
0C75C6  6100 FAA8       BSR UPREAD                  * DANN WERT LESEN
0C75CA  0201 000F       AND.B #$F,D1
0C75CE  0C00 0005       CMP.B #5,D0
0C75D2  6600 0006       BNE EGETLP1                 * WENN STD. ZEHNER, BIT 3 RUECKSETZEN WEGEN 24H
0C75D6  0201 0003       ANDI.B #$3,D1
0C75DA
0C75DA                  EGETLP1:
0C75DA  3401            MOVE D1,D2                  * UND RETTEN
0C75DC  0C00 0006       CMP.B #6,D0                 * TESTEN DB WOCHENTAG
0C75E0  6700 0022       BEQ EGETWT
0C75E4  1019            MOVE.B (A1)+,D0             * NAECHTE ADRESSE HOLEN
0C75E6  6100 FA88       BSR UPREAD                  * UND HOLEN
0C75EA
0C75EA  0201 000F       ANDI.B #$F,D1
0C75EE  3001            MOVE D1,D0                  * EINER IN D0
0C75F0  3202            MOVE D2,D1
0C75F2
0C75F2  6100 0078       BSR HEXBCD                  * UND IN BCD-ZAHL WANDELN
0C75F6  10C0            MOVE.B D0,(A0)+             * DANN DEN BCD-WERT IN PUFFER
0C75FB  0C11 00FF       CMP.B #$FF,(A1)             * ENDE
0C75FC  6700 000C       BEQ EGETEND
0C7600  6000 FFC0       BRA EGETLP
0C7604
0C7604                  EGETWT:
0C7604  10C1            MOVE.B D1,(A0)+             * WOCHENTAG BEDARF KEINER WANDLUNG
0C7606  6000 FFBA       BRA EGETLP                  * DANN WEITER
0C760A
0C760A                  EGETEND:
0C760A                  ESETEND:
0C760A  4CDF 03FF       MOVEM.L (A7)+,A0-A1/D0-D7
0C760E  4E75            RTS
0C7610
0C7610                  *** PROGRAMM UEBERTRAEGT DIE BCD-DATEN AUS DURCH A0 ADRESSIERTEN PUFFER WIE DIE
0C7610                  *   GRUNDPROGRAMMROUTINE @SETUHR
0C7610
0C7610                  ESETUHR:
0C7610  48E7 FFC0       MOVEM.L A0-A1/D0-D7,-(A7)
0C7614  43FA 005C       LEA EMUTAB(PC),A1
0C7618
```

```
0C7618                  ESETLP:
0C7618  1018            MOVE.B (A0)+,D0             * WERT AUS PUFFER IN D0
0C761A  0C11 0006       CMP.B #6,(A1)               * TESTEN DB WOCHENTAG
0C761E  6700 0032       BEQ ESETWT                  * DANN ANDERE FUNKTION
0C7622  6100 003A       BSR BCDHEX                  * NUN BCD-ZAHL IN 2 HEX-ZAHLEN WANDELN
0C7626  1400            MOVE.B D0,D2                * IN D0 SIND EINER, IN D1 ZEHNER
0C7628  1019            MOVE.B (A1)+,D0             * ADRESSE AUS TABELLE IN D0
0C762A  0C00 0005       CMP.B #5,D0
0C762E  6600 0006       BNE ESETLP1
0C7632  08C1 0003       BSET #3,D1                  * FUER 24 STD. UHR
0C7636
0C7636                  ESETLP1:
0C7636  2F02            MOVE.L D2,-(A7)             * D2 RETTEN, DA IN UPWRITE BENUTZT
0C7638  6100 FA46       BSR UPWRITE
0C763C  241F            MOVE.L (A7)+,D2
0C763E  1019            MOVE.B (A1)+,D0             * NAECHSTE ADRESSE
0C7640  1202            MOVE.B D2,D1                * WERT IN D1
0C7642  6100 FA3C       BSR UPWRITE                 * SCHREIBEN
0C7646  0C10 00FF       CMP.B #$FF,(A0)             * $FF IST ENDEZEICHEN
0C764A  6700 FFBE       BEQ ESETEND
0C764E  6000 FFC8       BRA ESETLP
0C7652
0C7652                  ESETWT:                     * WOCHENTAG WIRD SO GESETZT, DA NUR EINSTELLIG
0C7652  1200            MOVE.B D0,D1
0C7654  1019            MOVE.B (A1)+,D0
0C7656  6100 FA28       BSR UPWRITE
0C765A  6000 FFBC       BRA ESETLP
0C765E
0C765E                  BCDHEX:                     * WANDELT BCD-ZAHL AUS D0 IN HEXZAHL
0C765E  3200            MOVE D0,D1                  * WERT AUCH IN D1
0C7660  0240 000F       AND #$0F,D0                 * EINER IN D0
0C7664  0241 00F0       AND #$F0,D1                 * ZEHNER IN D1
0C7668  E859            ROR #4,D1                   * UND NOCH ROTIEREN
0C766A  4E75            RTS
0C766C
0C766C                  HEXBCD:                     * WANDELT HEXZAHL AUS D0 + D1 IN BCD-ZAHL IN D0
0C766C  E919            ROL.B #4,D1
0C766E  8041            OR D1,D0
0C7670  4E75            RTS
0C7672
0C7672                  EMUTAB:
0C7672  05 04 03 02 08  DC.B $5,$4,$3,$2,$B,$7,$A,$9,$C,$B,$6,$1,$0,$FF
0C7677  07 0A 09 0C 0B
0C767C  06 01 00 FF
0C7680
0C7680                  DS 0
0C7680
0C7680                  ***************
0C7680                  ******* HIER TESTPROGRAMME UND BEISPIELE
0C7680                  ***************
0C7680
0C7680                  *** TESTROUTINEN FUER AUFRUF UEBER TRAP
0C7680
0C7680                  TEST:
0C7680  6100 0052       BSR TRAPFIND                * USERTRAP INITIALISIEREN
0C7684  3E3C 02BE       MOVE #702,D7                * CODE IN D7
0C7688  363C 0001       MOVE #1,D3                  * AUSGABE, DAZU CODE IN D3
0C768C  41ED 0FCA       LEA OUT(A5),A0              * UND PUFFERADRESSE IN A0
0C7690  203C 0000003B   MOVE.L #$3B,D0              * KOORDINATEN BELEGEN
0C7696  243C 0000000A   MOVE.L #10,D2
0C769C  223C 0000000A   MOVE.L #10,D1
```

```
0C76A2  383C 0001       MOVE #1,D4                  * SEITENZAHLER VORBELEGEN
0C76A6  1C3C 003F       MOVE.B #INTDIS,D6
0C76AA  4E47            TRAP #7                     * AUFRUF NUN UEBER USERTRAP
0C76AC
0C76AC  3E3C 0010       MOVE #!CLR,D7
0C76B0  4E41            TRAP #1
0C76B2  3E3C 02C0       MOVE #704,D7
0C76B6  4E47            TRAP #7
0C76B8
0C76B8  3E3C 0010       MOVE #!CLR,D7
0C76BC  4E41            TRAP #1
0C76BE  3E3C 02BF       MOVE #703,D7
0C76C2  4284            CLR.L D4
0C76C4  4E47            TRAP #7
0C76C6  4E75            RTS
0C76C8
0C76C8                  *** TESTROUTINE FUER HEXREAD
0C76C8
0C76C8                  TEST1:
0C76C8  41F9 00005000   LEA $5000,A0
0C76CE  6100 FDC6       BSR HEXREAD
0C76D2  4E75            RTS
0C76D4
0C76D4                  *** PROGRAMM ZUM FINDEN UND STARTEN DER ROUTINE TRAPINIT
0C76D4
0C76D4                  TRAPFIND:                   * PROGRAMM SUCHT HEADER DER INITIALISIERUNGS-
0C76D4  41F9 00000400   LEA $400,A0                 * ROUTINE
0C76DA
0C76DA                  TRAPFLP:
0C76DA  B1FC FFFFFFFF   CMPA.L #$FFFFFFFF,A0        * WENN SPEICHER-ENDE ERREICHT
0C76E0  6700 001C       BEQ FLPERROR                * DANN FEHLER
0C76E4  0C58 55AA       CMP.W #$55AA,(A0)+          * SOLANGE BIS HEADER GEFUNDEN
0C76E8  6600 FFF0       BNE TRAPFLP
0C76EC  D1FC 00000002   ADDA.L #2,A0                * NUN PRUEFEN, OB REST AUCH OK
0C76F2  0C58 AA55       CMP.W #$2211AA55,(A0)+
0C76F6  6600 FFE2       BNE TRAPFLP                 * WEITER, WENN NICHT OK,
0C76FA  4E90            JSR (A0)                    * SONST AUSFUEHREN
0C76FC  4E75            RTS
0C76FE
0C76FE                  FLPERROR:                   * ALS FEHLER-CODE STEHT NUN $FFFFFFFF IN A0
0C76FE  4E75            RTS
0C7700
0C7700                  *** PROGRAMM ZUM FINDEN UND STARTEN DER ROUTINE INTINIT
0C7700
0C7700                  INTFIND:                    * PROGRAMM SUCHT HEADER DER INITIALISIERUNGS-
0C7700  41F9 00000400   LEA $400,A0                 * ROUTINE
0C7706
0C7706                  INTFLP:
0C7706  B1FC FFFFFFFF   CMPA.L #$FFFFFFFF,A0        * WENN SPEICHER-ENDE ERREICHT
0C770C  6700 FFF0       BEQ FLPERROR                * DANN FEHLER
0C7710  0C58 55AA       CMP.W #$55AA,(A0)+          * SOLANGE BIS HEADER GEFUNDEN
0C7714  6600 FFF0       BNE INTFLP
0C7718  0C98 1122AA55   CMP.L #$1122AA55,(A0)+
0C771E  6600 FFE6       BNE INTFLP
0C7722  4E90            JSR (A0)
0C7724  4E75            RTS
0C7726
0C7726                  ********
0C7726                  *** ROUTINE ZUM IMPLEMENTIEREN IN GRUNDPROGRAMM
0C7726                  ********
0C7726
0C7726                  * DIE ROUTINEN KOENNEN AUF FOLGENDE WEISE IN DAS GRUNDPROGRAMM IMPLEMENTIERT
0C7726                  * WERDEN :
0C7726                  * 1. GRUNDPROGRAMM INS RAM VERSCHIEBEN (Z.B. $20000
0C7726                  * 2. GRUNDPROGRAMM NEU STARTEN (MENUE = 2 PROGRAMM STARTEN ADR. $233F4)
0C7726                  * 3. RTC ROUTINEN LADEN UND ASSEMBLIEREN
0C7726                  * 4. GRUNDPROGRAMMEPROM 3 NEU BRENNEN (VON $24000 - $25FFF)
0C7726                  * 5. RTC-ROUTINEN IN EPROM BRENNEN ( $A7000 - $A7FFF NACH $1000 !)
0C7726                  * 6. EPROM AUF ADRESSE $C6000 STECKEN (4.SOCKEL BANK C)
0C7726                  * 7. GRUNDPROGRAMM-EPROM TAUSCHEN
0C7726                  * DIE ADRESSEN KOENNEN NATUERLICH ANGEPASST WERDEN
0C7726
0E4CF0                  ORG @GETUHR
0E4CF0  4EF9 000C75B4   JMP EGETUHR
0E4CF6
0E4D2E                  ORG @SETUHR
0E4D2E  4EF9 000C7610   JMP ESETUHR
0E4D34
0E4D34                  ENDEA:
0E4D34
0E4D34                  END.

0E9132  Ende-Symboltabelle
```

```
Uhr setzen
Datenformat = Tg/St/Mi/Se/Da/Mo/Ja

Sa/12/35/00/20/09/86
```

Bild 5. Ein Beispiel für die Bedienung: Das Stellen der Uhr

REFON
Diese Routine schaltet die Referenzsignale auf die Datenleitungen. Damit läßt sich ebenfalls jede Minute ein Interrupt auslösen oder, durch Abfrage der Schnittstelle, eine Timerfunktion realisieren. *Bild 6* zeigt die genauen Zeiten.

ESETUHR
Mit diesem Programm wird die Routine §SETUHR des Grundprogramms emuliert. So kann die Uhr auf die gleiche Weise gestellt werden wie bei der Orginalkarte mit dem E05-16. Die Daten werden im BCD-Format in einen durch A0 adressierten Puffer gelegt und von der Routine in den Baustein geschrieben, wobei sie in die die für den RTC verständliche Form übertragen werden.

EGETUHR
Tag, Datum und Uhrzeit werden im BCD-Format in einen Puffer geschrieben; die Pufferadresse steht in A0. Da sich dieses Programm exakt wie die Grundprogramm-Routine §GETUHR verhält, können alle Programme, die für die Orginal-Uhrenkarte geschrieben wurden, auch mit dem RTC-58321 verwendet werden. Man muß dazu lediglich im EPROM die Sprünge auf die Orginalroutinen im Grundprogramm durch solche auf die neuen Unterprogramme ersetzen.

Da im Listing mit Kommentaren nicht gespart wurde, dürfte das Umschreiben für andere Rechner kein Problem sein. Man sollte auch hier mit den Elementar-Funktionen beginnen und die übrigen Programmteile darauf aufbauen.

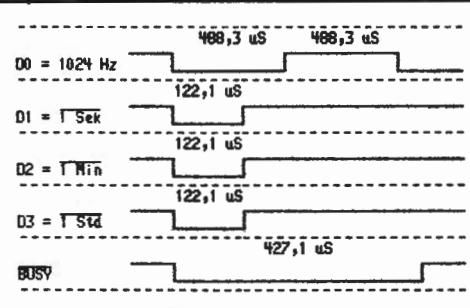


Bild 6. Diese Zeiten sind für den RTC-58321 spezifiziert

Kompetent und preiswert

Franzis Verlag · 8 München 37 · Postfach 37 02 80
Postvertriebsstück Gebühr bezahlt

Das schafft noc eder!

Spitzenprodukte der Microcomputertec logie zu kundenge. ten Preisen!

Commodore C64-II

Weltweit der Spitzenreiter in seiner Klasse! 64 K Speicher, Farbe Grafik, Sound ... Mit dem größten Software-Angebot, das es für einen Heimcomputer je gab.

Preissenkung nur noch **395.-**

C 64-A
wie C 64-II, jedoch im ehem. üblichen Standard-Gehäuse. Beide C 64 incl. GEOS-Diskette!! (Grafik-Benutzeroberfläche) 333.-

Noch mehr s d günstige Angebote:

HP 28 wissenschaftlich	.00	Atari 520 ST/M deutsch	649.00
HP 41 CV 2233 Schritte	9.00	LX 00 mit Interface VC 64	498.00
HP 41 CX + Time, X-Funct.	525.00	SEIKOSHA SP 100 VC	465.00
PC 1403 erw. auf 32 K Ram	299.00	SEIKOSHA SP 180 AI	465.00
CE 126 P Drucker	179.00	SL 00 AI 24-Nadel-Drucker	875.00
PB-220 G Basic 10 KB	119.00	CENTRONICS GLP	365.00
PB 1000 erw. auf 40 K	498.00	SEIKOSHA GP 100 AS	199.00
MD 100 3.5" Floppy Disk	698.00	SEIKOSHA GP 700 A Farb.	299.00
VC 1541 Floppy 170 K	399.00	NEC P 6	1100.00
VC 1551 Floppy für C16/+4	299.00	HIGHSCREEN Monitor Tuner	100.00
Maus f. C 64/66-II/128	79.00	KP 548 RGBI Stereo-Farbm.	895.00
MPS 1000 VC/PC-Interface	675.00	Highscreen DP 602 - P 31	159.00
1002 Farbmonitor	499.00	Highscreen DP 622 - LA	179.00
COMMODORE C 128	569.00	Highscreen DP 612 - P 39	100.00
VC 1571 Floppy für C 128	569.00	Highscreen DP 652 - Schw.	179.00
VC 1531 Datenrecorder	99.00	Highscreen DP 613 - TTL	199.00
AMIGA 500 Basispaket	1235.00	Highscreen KP 501 RGBI	479.00
AMIGA1080 Basispaket	1695.00	Disk. 5.25, 48 TPI control	9.95
SCRIBE-Drucker mit Kabel	248.00	Disk. 5.25, 1D15 braun, Karton	14.00
PC 10-II 640 K 2 LW	2145.00	Disk. 5.25, 2D15 blauer Karton	19.00
PC 10-II, 2 LW 20 MB	2875.00	Disk. 5.25, 1D25 schw. Karton	19.00
COMMODORE PC AT 40/20	4999.00	5.25, 1D25 Wendedisketten	22.00
20 MB incl. Cont. Schneider	799.00	5.25, 2D25 Nachfüllpack.	25.00
20 MB incl. Cont. Zenith	799.00	5.25, 2D25 Plastikbox	39.00
TA-PC 64 K RAM	169.00	5 Stck. 3.5 Zoll Disk. 100	19.00
Atari 520 ST/M 1 MB engl.	499.00	5 Stck. 3.5 Zoll Disk. 200	26.00

Commodore AMIGA 2000

Basismodell der AMIGA-Serie. Konzipiert als offenes System mit 5 AMIGA und 4 IBM-PC-Steckplätzen. Mit 2 Laufwerken (ohne Monitor)

AMIGA Monitor 1081 **775.-**

2695.-

Der JOYCE von Schneider

MEHR SCHREIBMASCHINE – WENIGER COMPUTER

Der große Vorteil: Speziell für Textverarbeitung konzipiert. Mit speziellen Tasten: Statt „CTRL-K + F" drücken Sie einfach die „SUCHE"-Taste zum Auffinden vom Textstellen.

TECHNISCHE DATEN:
- Z-80 Prozessor
- 256 K Speicher davon 112 K als schnelle Ram-DISK
- incl. 3" Laufwerk, 250 K
- CP/M 3.0, BASIC, Dr. Logo

Wir nehmen Ihre alte Schreibmaschine beim Kauf eines JOYCE-Textsystems für bis zu 150.- DM in Zahlung.

Komplett **1799.-**

Lieferumfang:
Incl. Monitor, Matrixdrucker NLQ
Textverarbeitungsprogramm LOCOSCRIPT,
DOS 3.0, BASIC, Dr LOGO

Wahlweise mit Typenraddrucker **2299.-**

SHARP PC 1260

Drucker CE 125 dazu **225.-**

erweitert auf 10 K **229.-**

Komplett **429.-**
Sie sparen 25.- DM im Vergleich zu den Einzelpreisen!

Commodore PLUS/4

64 K Speicher, 121 Farben, 4 Tongeneratoren, hervorragende Grafikmöglichkeiten. BASIC 3.5, 4 fertige Programme sind bereits eingebaut. Nur **199.-**

Ferguson Monitor 25 MHz, grün **129.-**

gelb **139.-**

Komplett mit grünem Monitor nur **259.-**

CITIZEN

Matrix-Drucker MDP 560 40 Zeichen/Zeile

Kassettens, etc. Kompletter VC-Zeichensatz

Anschlußkabel für COMMODORE VC 20, C 16, C 116, PLUS/4, C 64 und C 128 nur **99.-**

GÜNSTIGE KOMPLETTKOMBINATIONEN:

(Rot in Klammern steht was Sie im Vergleich zur Summe der Einzelpreise sparen)

COMMODORE PLUS/4 + Recorder 1531 + Lernkurs	229.00 (49.00)
COMMODORE PLUS/4 + Floppy 1551	399.00 (99.00)
COMMODORE PLUS/4 + Floppy 1551 + grünem FERGUSON-Monitor	485.00 (112.00)
COMMODORE PLUS/4 + CITIZEN 2-farb Drucker	259.00 (39.00)
COMMODORE C 64-II + Floppy 1541 + Farbmonitor 1802 + GEOS + Maus	1295.00 (106.00)
COMMODORE 128 D + HIGHSCREEN Monitor DP 612	1195.00 (80.00)
COMMODORE AMIGA 500 + Monochr.-Monitor 612	1399.00 (35.00)
COMMODORE AMIGA 500 + Farbmonitor KP 548	1799.00 (179.00)
CASIO PB 220 Pocket-Computer + Drucker FP12-s	199.00 (19.00)
ZENITH Z 148 IBM-komp. + 20 MB-Festplatte	2648.00 (145.00)
ATARI 130 XE + Matrixdrucker ATARI 1029	495.00 (69.00)
ATARI 520 ST + ATARI-Monitor	985.00 (63.00)
SINCLAIR SPECTRUM PLUS + 3.5" Laufwerk 720 K	745.00 (102.00)
SINCLAIR QL + 3.5" Laufwerk 720 K	895.00 (102.00)
SINCLAIR SPECTRUM PLUS + SEIKOSHA-DRUCKER GP 50 S (32-Zeichen-Grafik-Drucker)	399.00 (25.00)

Ausfüllen, ausschneiden und ab die Post an VOBIS, Postfach 1778, 5100 Aachen

Ich will mehr wissen! Senden Sie mir
☐ Prospekte ☐ Preisliste

Meine Adresse: _____

Schneider PC

PC 1512 SD Mit GEM und IBM-kompatibel. Incl. Monitor 1 Laufwerk 320 K **1499.-**

Sensationspreis: PC 1512 SD mit nachträglich eingebauter 20 MB-Textplatte (SCSI) **2298.-**

Zusätzlich mit zweitem gerüstetem 2. Laufwerk **1777.-**

kompetent + preiswert

WICHTIG! Herstellerbedingte Lieferzeiten! Aufgrund erhöhter Nachfrage ist nicht immer alles sofort lieferbar!

VOBIS
Deutschlands umsatzgrößter Microcomputer-Spezialist

VERSANDZENTRALE:
Postfach 1778
Rotter Bruch 32/34
5100 AACHEN
☎ 0241/50 00 81
Tx 832389 vobis d

Ab 4.7. auch in AUGSBURG, Jakoberstr. 16

FILIALEN:

AACHEN Viktorstr. 74 · 0241/54 31 00
BERLIN 30 Kurfürstenstr. 101 · 030/2 13 94 80
BIELEFELD Herforder Str. 106 · 0521/6 38 78
BREMEN Violenstraße 37 · 0421/32 04 20
DORTMUND Hamburger Str. 110 · 0231/57 30 72
DÜSSELDORF Wielandstr. 21 · 0211/35 99 64
ESSEN Huyssenallee 3 · 02/1/23 17 74
FRANKFURT Frankenallee 207/209 · 069/73 40 49
HAMBURG Krohnskamp 15 · 040/2 79 46 76
HANNOVER Berliner Allee 47 · 0511/81 65 71
KARLSRUHE Kriegsstr. 27/29 (am BGH) 0721/37 82
KONSTANZ Kreuzlinger Str. 18 · 07531/1 55 60
KÖLN Mathiastr. 24-26 · 0221/24 86 42
MÜNCHEN Abelestr. 3 · 089/77 21 10
NÜRNBERG Vordere Ledergasse 8 · 0911/23 29 95
STUTTGART Marienstr. 11-13 · 0711/60 63 36